# HYBRID SYSTEM FOR BOTANY RECOGNITION


NAME : YEW KOK WENG
MATRIX NUMBER : WEK 020295
SUPERVISOR : MR. MOHD NOOR RIDZUAN DAUD
MODERATOR : MS. JORIAH NORFIZLINA ISMAIL

FACULTY COMPUTER SCIENCE &
INFORMATION TECHNOLOGY

# ABSTRACT

Hybrid System for Botany Recognition is a system that aims to be use as a botany expert system to aid students or other users of botanical field on plant identification. It contain a leaf recognition module where the system will recognized a leaf template image and identify the features, a fuzzy logic module that refine the recognition module's output data and an image database consist of the detail of species build based on Case Based Reasoning concept.

This project proposal contains mainly the leaf recognition module. It contains a study of the current recognition technology, including both image processing field and the neural network field. It will also contain a brief review of the technology in discussion and configurations, both fit into the literature review section. System analysis and design will detail the overall system including the software and hardware involve. A section of methodology will review the method used based on the principle of software engineering and will be applied throughout the phase to aid a smooth development.

# ACKNOWLEDGEMENTS

This proposal is done for completing the course subject WXES3181 and WXES3182, the final year project. Acknowledgements are given to my supervisor, Mohd Noor Ridzuan Daud and my moderators, Prof. Madya Dr. Roziati Zainuddin and Ms. Joriah Norfizlina Ismail for freeing some time moderating and supervising this project. It's been an opportunity for being in a group with Mr. Wong Lig Ging. Valuable opinion and suggestion are greatly appreciated. Also gratitude is given to those listed in the reference for sharing their thoughts over the internet.

# TABLE OF CONTENT

# LIST OF TABLES

# LIST OF FIGURES

# CHAPTER 1: INTRODUCTION

## *1.1 Project Definition*

Hybrid System for Botany Recognition is standard execution system tend to help users of botanical field on the identification of tree species based on the leaf feature. The system is build up by 3 modules, which is:

1) **Leaf Recognition Module**

   This module intends to extract the features of the leaf query image. It analysis processes the query image; extract the features and classify the features extracted to predefined feature category. The output is then used as an input to match with the predefined species images in the build in image library to identify the query image's leaf species.

2) **Fuzzy Logic Module**

   This module provides a simple way to arrive at a definite conclusion based upon vague, ambiguous, imprecise, noisy, or missing input information. Fuzzy Logic's approach to control problems mimics how a person would make decisions, only much faster.

3) **Case Base Reasoning (CBR) Module**

   This module contains the leaf information database, including the images of the species. The species of the unknown leaf identified from the previous module will have its information retrieve from this module and display to the user. The user also can use the search engine to retrieve the species information if the digital leaf image is unavailable.

This proposal dealt with the 1st module: **Leaf Recognition Module**, while the other 2 module are handled by another 2 of group members. For the

**Leaf Recognition Module**, the system tend to recognize the unknown leaf based on the features such as shape, color or texture by using image analysis and neural network technology. The features extracted from the query image is then used as input query for the 3rd module by searching through the build in image database and find the match feature, retrieve the species of the unknown leaf along with the information related to the species. If the recognition module tends to return vague, imprecise result, the fuzzy logic module will then take over the process of identification by filtering the result and make decision as human does.

This system would act as a botany expert that would let the user to quickly identify tree species by providing a digital leaf image. However other user can use the system to keep their plant database and plant information. The similar framework of system can be modified so that it can be stored as digital face image recognition or other image analysis task that uses such modules.

## *1.2 Objectives*

The objectives of the project are:

- ✓ To build a system that would help the botanical field user to quickly identify plant species
- ✓ To ease the species information retrieval process
- ✓ To serve as a botany expert that could be access anytime
- ✓ To serve as a more flexible expert system that has a digital recognition ability

## *1.3 Scopes & Limitations*

### *1.3.1 Project Scopes*

The project will be developed within the parameter of:

- ✓ Develop a stand-alone system that can extract specific features of digital leaf images.
- ✓ Develop a system that can identify the species of the unknown leaf species within its knowledge base based on the feature.
- ✓ Information of the identified species will be retrieved from knowledge base and present to the users
- ✓ Develop a system that able to elicit useful information about a leaf from the user and based on the information provided, categorize it into its most belonging species.

While this proposal is solely discuss on the **Leaf Recognition Module**, this module will be developed based on the following parameter:

- ✓ Develop a recognition engine that would recognize the unknown leaf digital image
- ✓ Develop a recognition engine that matches human recognition ability that is recognition not by exact match but by similarity
- ✓ The system would have an efficient leaf knowledge elicitation process that ease the users on adding new species
- ✓ Provide links and parameters to other modules

### *1.3.2 Limitations*

Apart from the scope, this system has limitation whereby:

- Optimize only for botanical field leaf type.
- Recognition is only depends on a few features of the leaf such as shape, color or structure.
- Retrieval process extensively consumes computational power.
- Information elicitation process is not as flexible as human experts.
- Knowledge base only stores knowledge of experts, but not experience.

This module is restricted by following limitations:

- Decrease in recognize accuracy with the digital image contains noises or too many contents.
- Recognition error would lead to false species information retrieval presented to users.
- Recognition process extensively consume computational power

## *1.4 Aims*

This system is aimed to:

- ✓ Provide a simple recognizing and information retrieving engine that ease the use of users
- ✓ Provide a knowledge base retrieval system that would match up a botanical experts or encyclopedias
- ✓ Enable fast but high accuracy retrieval of information

Apart from that, the **Leaf Recognition Module** is aimed to:

- ✓ Develop an efficient algorithm to extract features of leaf for matching process.
- ✓ Develop an image retrieval algorithm that fits not only leaf but any digital images.
- ✓ Develop an algorithm that results in high retrieval accuracy.

## *1.5 Expected Outcome*

The following would be expected from the system:

- ✓ Ease of use user interfaces that fulfill the user-friendliness requirement.
- ✓ Acceptable response time between data retrieval and matching process

And for the Leaf Recognition Module, the following outcome is expected:

- ✓ An acceptable response time and error rate of retrieval process.
- ✓ A constant accuracy

## 1.6 *Project Schedule*

To clarify the time management and smooth the control, a project schedule that consists of the whole development's activities is essential to the developer to achieve a systematic progress and ensure on-time delivery of the product. This project is scheduled as below:

| | Task Name | Duration | Start | Finish |
|---|---|---|---|---|
| 1 | Introduction | 7 days | Mon 7/5/04 | Tue 7/13/04 |
| 2 | Milestone | 0 days | Tue 7/13/04 | Tue 7/13/04 |
| 3 | Literature Review | 14 days | Wed 7/14/04 | Mon 8/2/04 |
| 4 | Milestone | 0 days | Mon 8/2/04 | Mon 8/2/04 |
| 5 | Methodology | 14 days | Tue 8/3/04 | Fri 8/20/04 |
| 6 | Milestone | 0 days | Fri 8/20/04 | Fri 8/20/04 |
| 7 | System Analysis | 7 days | Mon 8/23/04 | Tue 8/31/04 |
| 8 | Milestone | 0 days | Tue 8/31/04 | Tue 8/31/04 |
| 9 | System Design | 14 days | Wed 9/1/04 | Tue 9/14/04 |
| 10 | Milestone | 0 days | Tue 9/14/04 | Tue 9/14/04 |
| 11 | System Implementation | 75 days | Wed 9/15/04 | Wed 12/22/04 |
| 12 | Milestone | 0 days | Wed 12/22/04 | Wed 12/22/04 |
| 13 | System Testing | 20 days | Thu 12/23/04 | Wed 1/19/05 |
| 14 | Milestone | 0 days | Wed 1/19/05 | Wed 1/19/05 |
| 15 | System Evaluation And Con | 20 days | Thu 1/20/05 | Wed 2/16/05 |
| 16 | Milestone | 0 days | Wed 2/16/05 | Wed 2/16/05 |
| 17 | Ducumentation | 171 days | Mon 7/5/04 | Wed 2/16/05 |
| 18 | Milestone | 0 days | Wed 2/16/05 | Wed 2/16/05 |

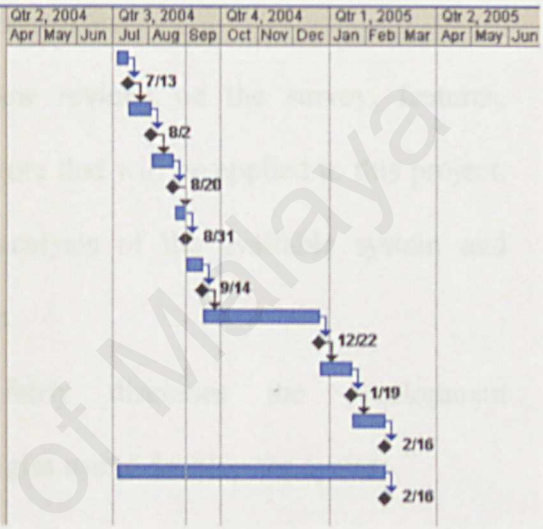



**Figure 1.1: Project Schedule**

## 1.7 Chapter Summary

This report is divided into 8 chapters. A brief synopsis of each chapter is as below:

- *Chapter 1* *Introduction* serves as an introduction to the entire project. It overviews the project's objectives, scopes, system, aim, expected outcomes, and project schedules.
- *Chapter 2* *Literature Review* reviews on the survey, features, capabilities and system architecture that will be applied to this project. It contains the research and analysis of the available system and techniques on the project domain
- *Chapter 3* *Methodology* fairly discusses the development methodology, tools and technologies use to develop the system.
- *Chapter 4* *System Analysis* discusses the requirements of the system including the functional and the non-functional as well as hardware and software requirements
- *Chapter 5* *System Design* documents the aspects that build up a system. It contains the functions that have to be implemented by the system including screen or interface design, data flowing and all modules that used in the system.
- *Chapter 6* *System Implementation* review the system development process that convert the modules and algorithm that have been design into instructions that can be implemented using certain computer programming language.
- *Chapter 7* *System Testing* document the work done to ensure the system function as stated in the requirements and specifications. .

- *Chapter 8 System Evaluation and Conclusion* reviews the outcome of the system as well as the problems arise and the solutions best suit. It also discuss the advantages and disadvantages of the system, enhancement that can be done and conclude the project.

# CHAPTER 2: LITERATURE REVIEW

## *2.1 Introduction*

For the **Leaf Recognition Module**, the overall recognition process can be divided into several processes. The digital images will firstly being analyzed according to predefined algorithm to identify the feature of each species. The features is then stored together with the digital image in the image library. For the recognition process, when the system is supplied with a query image, the image is then being analyzed with the same algorithm, and the feature be classified to the predefined class and will serve as an input for matching similarity with the features of the stored images in the CBR database.

The literature review conducted will focus on the method that can be applied to this system. Most of the technologies come from the Image Processing and Artificial Neural Network field. In section 2.2, the major techniques of image retrieval will be briefly introduced. This is following by the domain study in section 2.3 with a more depth of the techniques introduce with section 2.4 showing a few current system that uses the technologies. In section 2.5, the technologies considered to be use is being discussed.

## 2.2 *Technique Review*

### 2.2.1 *Content-Based Image Retrieval*

There is a wealth of research done in developing image retrieval techniques. For the image processing technique, it can be classified into three categories: *text-based retrieval*, *content-based retrieval*, and *semantic based retrieval*. Text-based retrieval techniques attempt to manually define description of the image, and image access is done through keyword matching. The performance is limited by the set of usually incomplete keywords maintained in the system, since an image is semantically richer than text descriptions. While the natural language used to associate the text with the image's content, it will result in ambiguity where interpretations by different people result in different text descriptors.

In content-based retrieval technique, the objects, color, texture, and shape extracted from an image is used as the basis for retrieval. However, these techniques are limited by the difficulty in specifying abstract queries. In the semantic-based retrieval technique, semantic meanings are used to retrieve relevant images. Typically, some form of knowledge base is required in the semantic-based retrieval systems. Content-based techniques are suitable for large image databases because they are able to perform automatic content extraction.

Content-based retrieval techniques currently can be classified into *template matching*, *global features matching*, and *local features matching*. In template-matching techniques, templates are used to detect objects within an image. The effectiveness of this technique is greatly affected by image noise,

and the orientation of the template. Global features such as color, texture or shape information have also been widely used to retrieve images. Shape retrieval technique usually works in specialized application domains that have very distinct shape. Color and texture are more suitable for general purpose application domains.

### 2.2.2 *Artificial Neural Network*

Another field that becomes more and more popular in solving problems such as mapping, clustering and constrained optimization is the artificial neural network (ANN). By understanding the human brain as a biological neural network, ANN takes the advantage of human way of learning and implement into such problem solving area.

Artificial Neural Network can be defined as an information-processing system that has certain performance characteristics in common with biological neural networks. A neural net consist of large number of simple processing elements called neurons. Each neuron is connected with each other, resulting in a network of neurons that can be applied to solve variety of problem, such as storing and retrieving data or patterns, grouping similar patterns or finding solutions to constrained optimization problems.

A neural network is characterized by (1) architecture, (2) training or learning algorithm and (3) activation function. ANN architecture can be divided into single or multi layer net. A single layer net (Figure 2.2) has no hidden layer between the input and output units whereas the multilayer net (Figure 2.3) has, as both are feed-forward nets. Figure 2.4 show a neural net that has closed-loop signal paths from a unit back to itself.

In addition to the architecture, the learning algorithm distinguishes the characteristic of different neural net. Learning algorithm is a method of setting the weights of the neural nets.

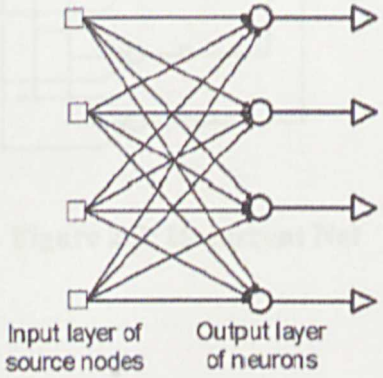



**Figure 2.1: Single Layer Net**

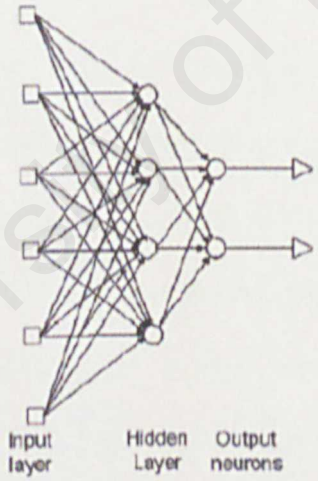



**Figure 2.2: Multi Layer Net**

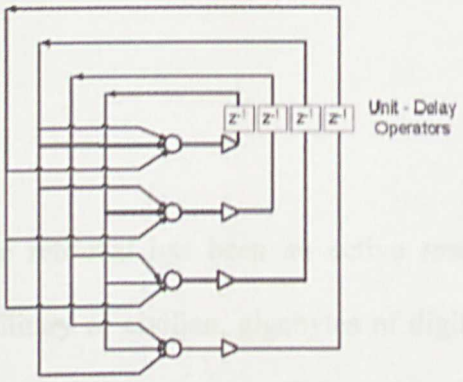



**Figure 2.3: Recurrent Net**

## 2.3 Domain Study

### 2.3.1 Introduction

Digital image retrieval has been an active research area since the 1970's. Either by military or civilian, gigabytes of digital image library has encouraged the development of the efficient browsing, searching and retrieving of the digital image [1]. Emergence of large-scale image collection causes technique applied became less and less effective. Many techniques have been proposed, each managing the weaknesses of each other so that the best approach would be implemented with the available hardware technology.

#### 2.3.1.1 Content Based Image Retrieval

*Content-based image retrieval* is defined as the retrieval of relevant images from an image database based on automatically derived imagery features [9]. Content based image retrieval was proposed to overcome the difficulty of traditional text-based image retrieval paradigm. Large size of image collection, rich content in images and subjectivity of human perceptions had encourage the image retrieval system being built using the new content based paradigm. In this section, only the basis of content based image retrieval will be discuss.

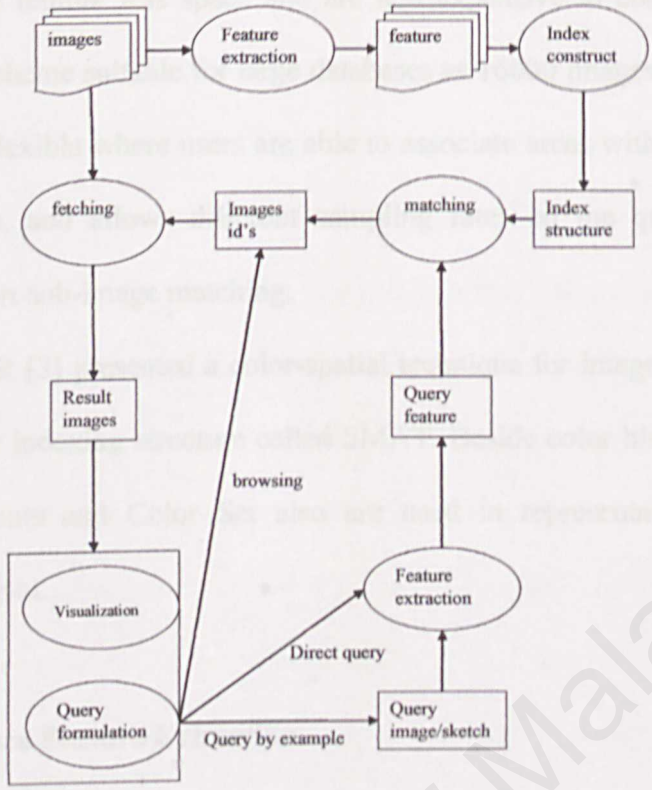



**Figure 2.4: Content-based image retrieval framework**

- **Color Feature Extraction**

Color feature is the most widely used feature extraction in the image retrieval process. One of the common methods used is the color histogram, where it takes into account the probability of intensity of the three color channels.

SamMatch [2] capture the contents of the regions of interest and used to compute similarity. SamMatch is inspired by the digitization of sound. It samples an image with respect to space interval. For each sampling region, the average color of pixels in the region is computed. The result forms the feature vector of the image.

SamMatch requires less space compared to correlogram, multiple color histogram or color spatial information technique. The smaller feature

vector require less space and are less expensive to compare, making this scheme suitable for large databases as 16000 images. SamMatch is also flexible where users are able to associate areas with objects in the image, and allows different sampling rates on the query image to support sub-image matching.

VIPER [3] presented a color-spatial technique for image retrieval with a new indexing structure called SMAT. Beside color histogram, Color Moments and Color Set also are used in representation in Image Retrieval.

- **Texture Feature Extraction**

Texture refers to the visual patterns that have properties of homogeneity that do not result from the presence of only a single color or intensity [5]. Texture feature extraction does not take only a single color or intensity, but is an innate property of virtually all surfaces. It contains the important information about the structural arrangement of surfaces and their relationship to the surrounding environment. In early 90's, researchers began to study the use of wavelet transform in texture representation after it was introduced and its theoretical framework established.

WALRUS [6] employs a novel similarity model in which each image is first decomposed into its regions, and the similarity measure between a pair of images is then defined to be the fraction of the area of the two images covered by matching regions from the images. WALRUS's image similarity model is designed specifically to address shortcomings

for the cases when images contain .similar regions but the region in one image is a translation or a scaling of the matching region in the other. WALRUS effectively eliminates the scaling and translation problem, not only at the image level, but also at the object level. N. Vujovic and D. Brzakovic [7] assuming that the random pattern may subject to misregistration relative to its representation in the database and assuming that it have missing parts. Matching is archived based on the texture pattern obtained by randomly embedding short fibers into the document medium during fabrications. Multi-wavelets have performed well and prove to be able to solve problems of finding optimal corresponding points. It also shows that multi-wavelets perform better than scalar ones.

- **Shape**

Shape representation can be divided into boundary-based or region-based. The former uses only the outer boundary of the shape while the latter uses the entire shape region. The similarity approach in IRM [9] reduces the influence of inaccurate segmentation, helps to clarify the semantics of a particular region, and enables a simple querying interface for region-based image retrieval systems. It was more accurate and faster retrieval compared with existing algorithms IRM reduces the adverse effect of inaccurate segmentation, an important property that previous work has virtually overlooked. Farzin Mokhtarian and Sadegh Abbasi [10] [11] address the problem in two-dimensional shape representation and matching in presence of self-intersection. Combine

with a modified conventional matching algorithm, an effective shape representation method has been proposed by describing a shape using maxima and minima of its Curvature Scale Space (CSS) contours. Daniel Curtis and David A. Meyer [12] proposed that Fourier transform is qualified as a superior quantum algorithmic solution for locating a template as a subimage for a larger image.

#### 2.3.1.2 *Artificial Neural Network*

Neural network methodology is replacing many traditional tools in the field of knowledge discovery and some related fields as the computers become faster and faster. For example, speech recognition neural networks software can now work on almost any home computer. Beyond doubt, the image recognition is going to be next. [12] tested possible application of neural network technology in artificial vision. [13] investigate the ability of a neural network to classify real world objects by implementing a multi-layer perceptron. NNFIR [15] is a human-computer interaction approach to CBIR using Radial Basis Function network. NNFIR firstly extract the different feature classes of query image. The system then compares the features of query image with the images stores in database, resulting in a group of metric value vectors based on individual feature classes. Then, the system combines the metric value vectors obtained from individual feature classes based on their importance to form the final set of retrieved images. If the user is not satisfied with the result, he marks the retrieved images with '*alike*', '*similar*', or '*different*' tags. Using the tags, the system

incrementally refines the radial basis function network, and searches again. NNFIR is also able to learn from user interaction, allowing incremental learning by relevance feedback. Chih-Fong Tsai propose a two-stage mapping model (TSMM) [16] which the idea is to design two level-0 generalizer to classify color and texture features into color and texture concept. Then a level-1 generalizer is designed to classify the color and texture concepts as middle level concepts into high level conceptual classes.

[17] futher improve the model by presenting a new approach by classify images based on the combination of image processing technique and neural networks. An image is assign with multiple keywords to represent its semantic contents. Then the images are divided into number of regions and color and texture features are extracted. One of the classifier, a self-organizing map (SOM) clusters similar images based on the extracted features. Then these clusters are labeled and used to train the second classifier which consist of several support vector machines (SVM).

## *2.4 Current Applications*

### *2.4.1 Analysis of Existing System*

**I. Photobook**

- **URL**

  http://vismod.www.media.mit.edu/vismod/demos/photobook/index.html.

- **Features**

  Photobook implement three different approaches to represent different type of image content: faces, 2D shapes and texture image. The first two uses eigenvectors of a covariance matrix as an orthogonal coordinate system of image space. As for texture, silhouette is extracted and a number of feature points on this are chosen. Then these feature points are use as nodes in building finite element model of shape. There is grid of still images displayed where user can selects some images or user can enters an annotation filter. From the result, user can select another query images and loop the search. Images are sorted by similarity with the query images and presented to user.

- **Applications**

  The technology of photobook has been used by Visage Technology in a FaceID package, which is used in several police departments.

Result of query: vistex sar Buildings.0005.3

Click on an image to make a new query, or

Randomize
Change the number of images to display
5 10 15 20
Change the similarity metric (which image features to use)
ohta rhist sar
Change the database
vistex feret faces

**Figure 2.5: Photobook result**

## II. QBIC

- **URL**

  http://wwwqbic.almaden.ibm.com/

- **Features**

  The texture features used in QBIC are modified versions of the coarseness, contrast, and directionality features proposed by Tamura. The shape features consist of shape area, circularity, eccentricity, major axis orientation and a set of algebraic moment invariants. The major axis orientation and the eccentricity are computed from the second order covariance matrix of the boundary pixels: the major axis orientation as the direction of the largest eigenvector and eccentricity as the ratio of the smallest eigenvalue to the largest one. For the database images, these shape features are extracted for all the object contours, semiautomatically computed in the database population step. In this process, the user enters an approximate object outline, which is automatically aligned with the nearby image edges, using the active contours technique. In this object identification step, the user can also associate text to the outlined objects.

- **Applications**

  The QBIC is applied at the server URL above.


**Figure 2.6: QBIC**

## III. Blobworld

- **URL**

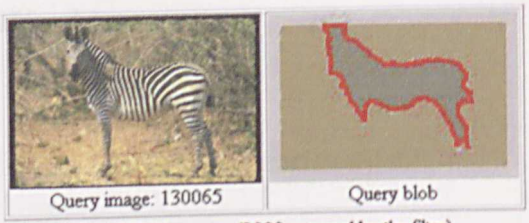
  http://elib.cs.berkeley.edu/photos/blobworld/.

- **Features**

  The color, texture, location and shape of regions (blobs) and of the background are used for querying. The color is represented by a histogram of 218 bins of the color coordinates in lab-space. Texture is represented by mean contrast and anisotropy over the region, as the 2D coordinate (contrast, contrast x anisotropy). Shape is represented by (approximate) area, eccentricity, and orientation. The user first selects a category in the search space. Then the user selects a region (blob) in an initial image indicating the importance of the blob. Next, the user indicates the importance of the blob's color, texture location and shape. The quadratic form distance is used to match two color histograms. The retrieved images are ranked in linear order, presented together with the segmented version showing the regions.

- **Applications**

  The demo on the web provides retrieval from a collection of 10000 Corel stock photos.




| | blob and feature importance: | | | | |
|---|---|---|---|---|---|
| | blob (overall) | color | texture | location | shape |
| blob 2 | very | not | not | not | very |

Querying from 35000 images (2000 returned by the filter).

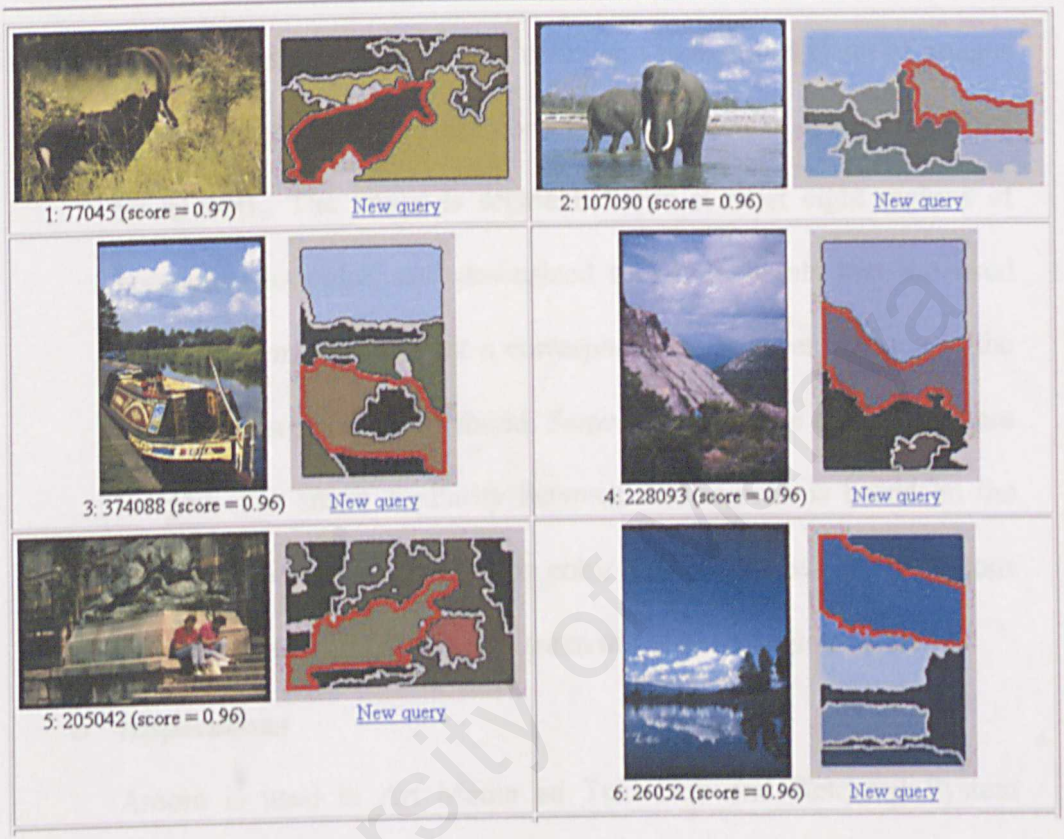



**Figure 2.7: Blobworld result**

## IV. Amore

- **URL**

  http://www.ccrl.com/amore/.

- **Features**

  The user first select a category of images and an initial set of images selected at random or by keyword. Query image can also be specified by its URL. The image is segmented into at most eight regions of homogeneous color, and downsized to 24x24 pixels that are used directly for matching. First a correspondence between regions in the query and target image is found. Same regions in the other image are merged. The shape similarity between two regions is based on the number of pixels of overlap. The color similarity between two regions is the distance in HLS space between the uniform region colors

- **Applications**

  Amore is used in Art Media ad Text Hub and Retrieval System (http://www.isi.edu/cct/arthur/),

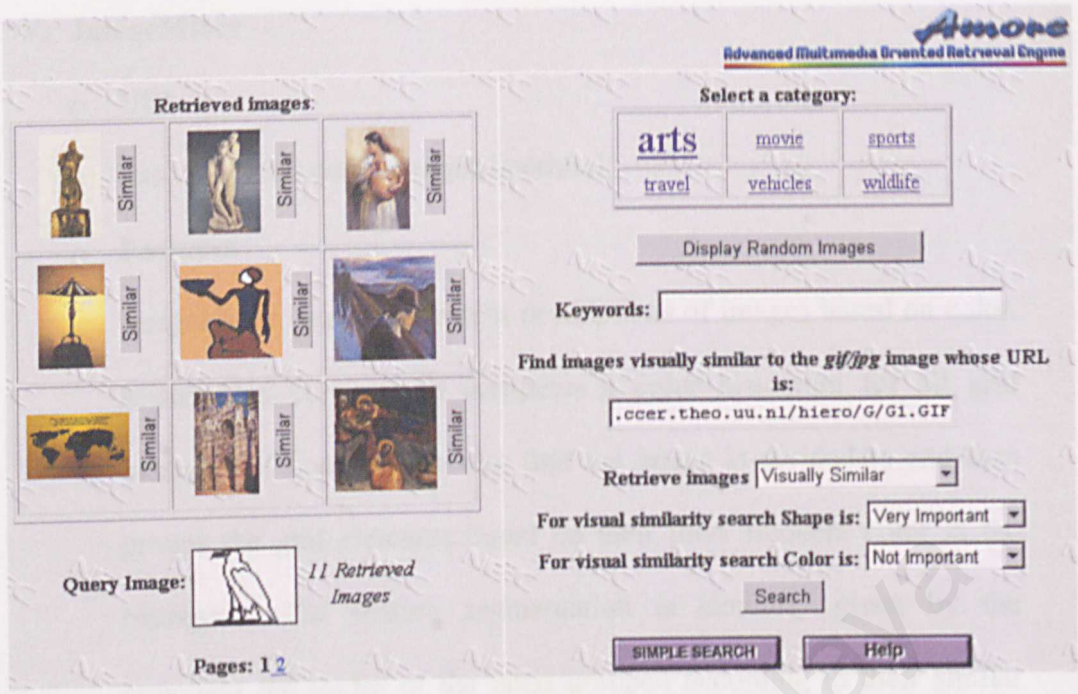



**Figure 2.8: Amore result**

## V. ImageMiner

- **URL**

  http://www.tzi.de/bv/ImageMinerhtml/.

- **Features**

  ImageMiner generates content descriptions of images based on color, texture and contours. It computes a color histogram for all grid elements of homogenous size that the image is divided to and then groups the grid elements based on their most frequent color in the histogram. The texture segmentation is similarly given by the bounding rectangles of the grids grouped according to their similar texture. For representing texture, a statistical approach is adopted. The contour-based shape description is obtained by first detecting the edge points using a gradient method and then connecting these points to form contours. Shape parameters are extracted for the closed regions. The relations of the contours and the color and texture regions are represented in a graph. Graph parsing algorithms classify scenes and generate a description with keywords and values. The user uses SQL like queries with keywords and values. The retrieval is performed by the IBM Search Manager text searcher.

- **Applications**

  Videos are made available for retrieval with ImageMiner.

## 2.5 *Technology*

### 2.5.1 *Image Analysis*

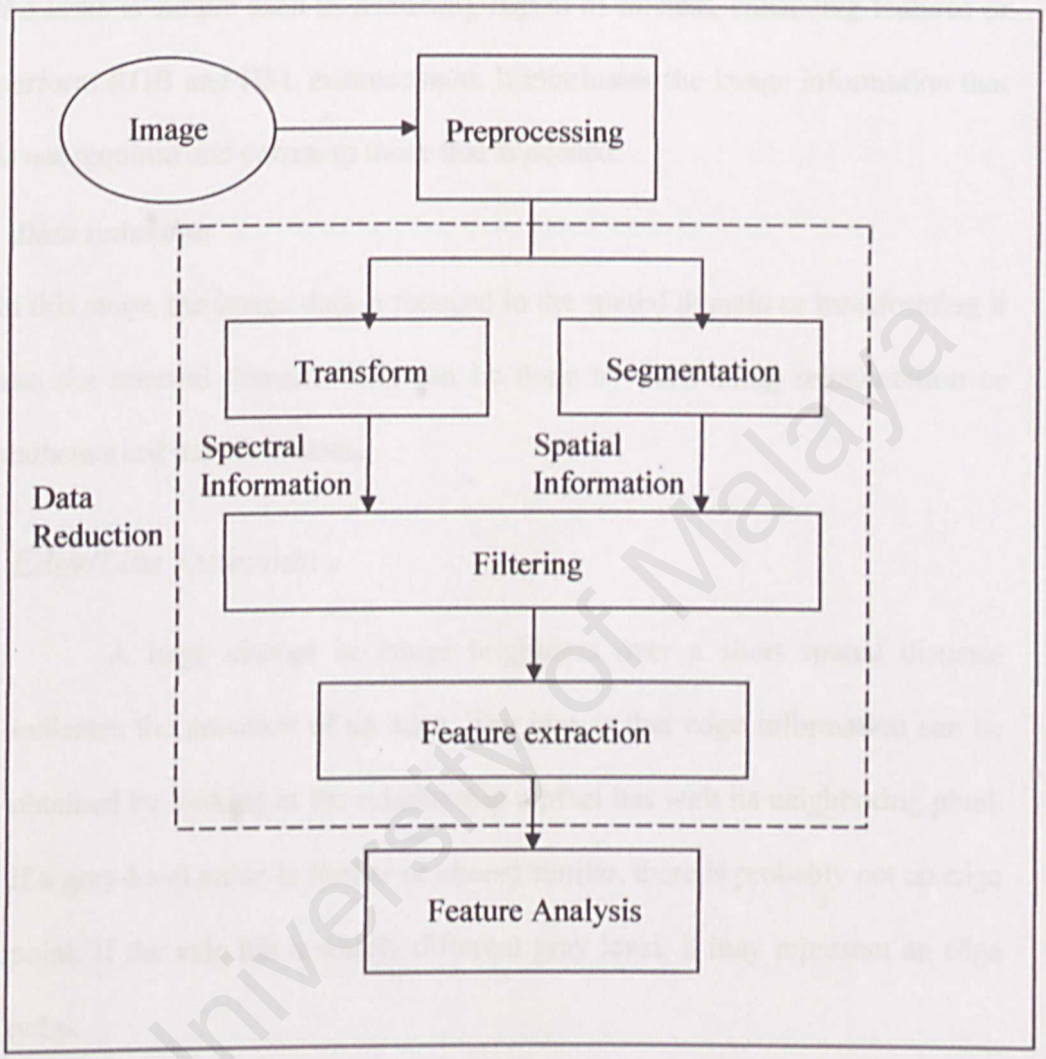



**Figure 2.9: Image Analysis** [22]

Image analysis is primarily a data reduction process. To solve an imaging problem, megabytes of image data often will be reduce to the information that is necessary for the process. In this project, image analysis is used to determine the process required to extract the specific parameters for the classification process. Figure 2.1 show the break down of the image analysis process:

- ***Preprocessing***

The preprocessing technique is to perform initial processing on the digital image so that it makes the following process much easier. It is a stage where the tasks is simple such as extracting region of interest, enhancing features or perform RGB and HSL enhancement. It eliminates the image information that is not required and enhances those that is needed.

- ***Data reduction***

In this stage, the image data is reduced in the spatial domain or transforming it into the spectral domain. This can be done by performing segmentation or mathematical transformation.

### 2.5.2 *Edge/Line Detection*

A large change in image brightness over a short spatial distance indicates the presence of an edge. The idea is that edge information can be obtained by looking at the relationship a pixel has with its neighboring pixel. If a grey-level value is similar or almost similar, there is probably not an edge point. If the vale has a widely different gray level, it may represent an edge point.

Many edge and line detector are implemented using convolution mask. Convolution process is overlaying a mask on the image, multiply the coincident values and sum the results. Many of these operators are sensitive to noise which is due to light fluctuations, camera lens, etc. There is a tradeoff between sensitivity and the accuracy of an edge detector. If the parameter set so that the edge detector is sensitive, it may find many edge point that attributable to noise. If it is set to less sensitive, accuracy will drop and it may miss out some edges. So, it is important preprocess the image to

minimize the noise effects. Because of the computational complexity, Kirsch Compass Masks, Robinson Compass Masks and Frei-Chen Masks will not be introduced here.

- *Roberts Operator*

The Roberts operator is the simplest edge detector and work best on binary images. It marks edge points only without returning the edge orientation. There are two form of operators where first consist of square root of the sum differences of the diagonal neighbor squared, while another contain the sum of magnitude of the differences of the diagonal neighbor squared without the square root.

1st form = $\sqrt{[I(r,c)-I(r-1,c-1)]^2+[I(r,c-1)-I(r-1,c)]^2}$

2nd form = $|I(r,c)-I(r-1,c-1)|+|I(r,c-1)-I(r-1,c)|$

- *Sobel Operator*

Sobel Operator consists of two masks that look for edge point in both horizontal and vertical directions. So at each pixel location we have two value $s_1$ corresponds to the row mask and $s_2$ from the column mask. This information is then combined into a single metric. The value is then use to compute the edge magnitude and the edge direction.

$$S_1=\begin{bmatrix}-1 & -2 & -1\\ 0 & 0 & 0\\ 1 & 2 & 1\end{bmatrix} \quad S_2=\begin{bmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{bmatrix}$$

$$\text{Edge Magnitude} = \sqrt{S_1^2+S_2^2} \qquad \text{Edge Direction} = \tan^{-1}\left(\frac{S_1}{S_2}\right)$$

- *Prewitt Operator*

Prewitt Operator uses the concept similar to Sobel Operator, but different mask coefficient. With the convolution process with the two masks, each pixel location will result in: $p_1$ result from the row mask and $p_2$ from the column mask. Edge magnitude and direction can be with this combination of value.

$$P_1 = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \quad P_2 = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

$$\text{Edge Magnitude} = \sqrt{P_1^2 + P_2^2} \qquad \text{Edge Direction} = \tan^{-1}\left(\frac{P_1}{P_2}\right)$$

- *Laplacian Operators*

The Laplacian masks are rotationally symmetric, which mean edges from all orientations contribute to the result. They are applied by selecting one mask of the three and convolve with the image. The sign of the result from two adjacent pixel locations provides directional information, and tells which side of the edge is brighter.

If we are interested in the edge information, the sum of the coefficient should return zero. The larger the sum, the less the processed image will change from its original image.

$$L_0 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \quad L_1 = \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix} \quad L_2 = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

| | |
|---|---|
|  | Original image |
|  | Roberts Operator |
|  | Sobel Operator |

| | |
|---|---|
|  | Prewitt Operator |
|  | Laplacian operators |

**Table 2.1: Comparison on the result of various edge detection operators**

### 2.5.3 *Artificial Neural Network*

Artificial neural network are describe in terms of their architecture (patterns of connection) and in terms of the way they are trained (rules for modifying weights). The following will introduce briefly a few types of architecture and learning algorithm.

- *Perceptron*

  Perceptrons had the most far reaching impact of any early neural nets. The architecture of a Perceptron consists of a single input layer of many neurons, and a single output layer of many neurons. But to be called a Perceptron, the network must also implement the Perceptron learning rule for weight adjustment. This learning rule compares the actual network output to the desired network output to determine the new weights.

  For each training input, the net would calculate the response of the output unit. The net did not distinguish between an error which is calculated output was 0 and the target was -1, but it was if the output was +1 and the target was -1. The sign of an error denotes that the weights should be changed in the direction indicates by the target value according to the formula:

$$w_i(new) = w_i(old) + \alpha t x_i$$

where t is the target value, $\alpha$ is the learning rate. Training would continue until no error occurs or the training steps ends.

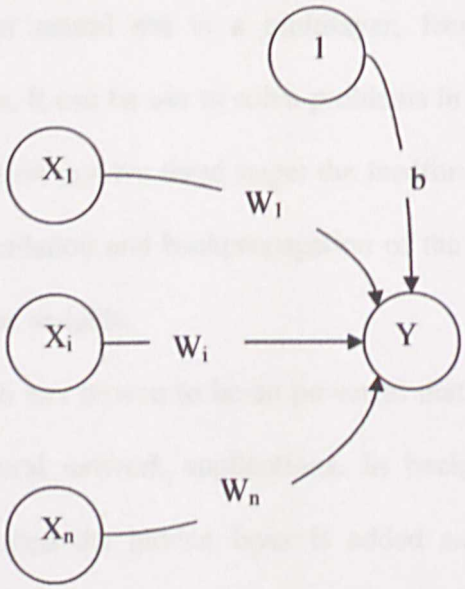



**Figure 2.10: Perceptron architecture**

- *ADALINE*

The ADALINE is a modification of the Perceptron, which typically uses bipolar (+1 or -1) activations for input signal, and add "bias" which activation is always +1. But the most important modification is the use of a **delta learning rule** or also known as least mean squares or Widrow-Hoff rule. As with the Perceptron, the delta rule compares desired output to actual output to compute weight adjustment. But the delta rule squares the errors and averages them to avoid negative errors canceling out positive ones.

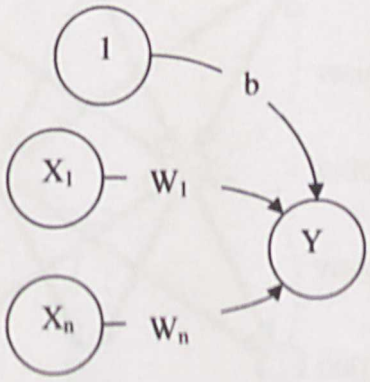



**Figure 2.11: ADALINE architecture**

- *Backpropagation Neural Net*

Backpropagation neural net is a multilayer, feedforward net trained by backpropagation. It can be use to solve problems in many areas. The training of backpropagation involve three stage: the feedforward of the input training pattern, the calculation and backpropagation of the associated error, and the adjustment of the weights.

Backpropagation has proven to be so powerful that it currently accounts for 80% of all neural network applications. In backpropagation net, a third neuron layer called the hidden layer is added and the discrete threshold function is replaced with a sigmoid (continuous) one. But the most important modification for backpropagation net is the **generalized delta rule**, which allows for adjustment of weights leading to the hidden layer neurons in addition to the usual adjustments to the weights leading to the output layer neurons. Using the generalize delta rule to adjust the weights leading to the hidden units is backpropagating the error-adjustment.

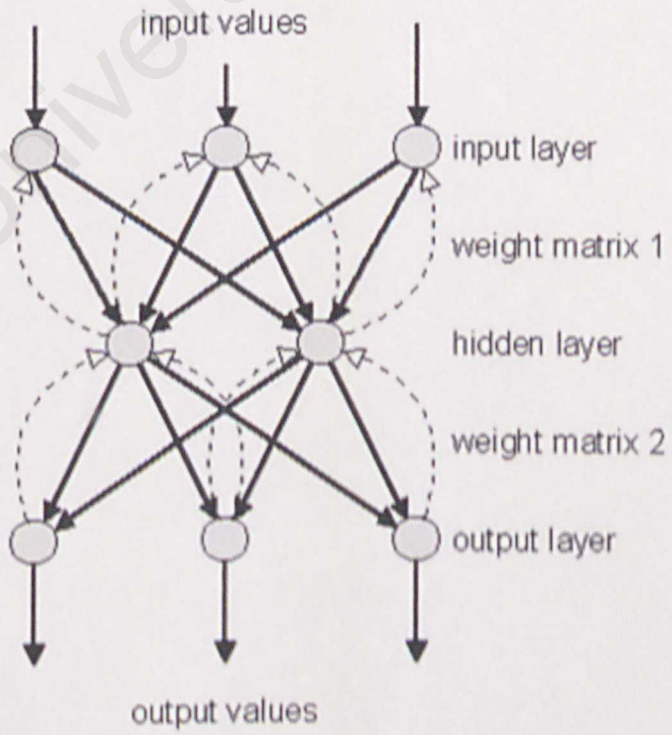


**Figure 2.12: Backpropagation net architecture**

# CHAPTER 3: METHODOLOGY

## *3.1 Justifications*

A methodology can be define as a set of activities, methods, best practices, deliverables and automated tools that system developers and project managers are use to develop and continuously improve information systems and software [23]. Well organized methodology helps the software developer to speed up and simplify the software development process, also helps the system developer to plan, manage, control and evaluate information system projects.

While this project is done by 3 people and that every one requires extensive research and development, the development process may be changes rapidly as time goes. Much of the work may need to be done iteratively. So a methodology that illustrates a clear event changes is needed.

### *3.1.1 Software Development Tools*

| Life-Cycle Model | Strengths | Weaknesses |
|---|---|---|
| Evolution-tree model (Section 2.2) | Closely models real-world software production<br>Equivalent to the iterative-and-incremental model | |
| Iterative-and-incremental model (Section 2.5) | Closely models real-world software production<br>Underlies the Unified Process | |
| Code-and-fix model (Section 2.9.1) | Fine for short programs that require no maintenance | Totally unsatisfactory for nontrivial programs |
| Waterfall model (Section 2.9.2) | Disciplined approach<br>Document driven | Delivered product may not meet client's needs |
| Rapid-prototyping model (Section 2.9.3) | Ensures that the delivered product meets the client's needs | Not yet proven beyond all doubt |
| Extreme programming (Section 2.9.4) | Works well when the client's requirements are vague | Appears to work on only small-scale projects |
| Synchronize-and-stabilize model (Section 2.9.5) | Future users' needs are met<br>Ensures that components can be successfully integrated | Has not been widely used other than at Microsoft |
| Spiral model (Section 2.9.6) | Risk driven | Can be used for only large-scale, in-house products<br>Developers have to be competent in risk analysis and risk resolution |

**Figure 3.1: Comparison of life cycle [24]**

Figure 3.1 briefly compares a few software life cycle models with its strength and weaknesses. From [24], Spiral, synchronize-and stabilize, extreme, rapid-prototype and code-and fix model are not suitable as the model fits more on large scale, less research system. While the waterfall model fails to show the flow of the system event, the evolution-tree model is used in this project as a software development methodology.

## 3.2 *Evolution-Tree Model*

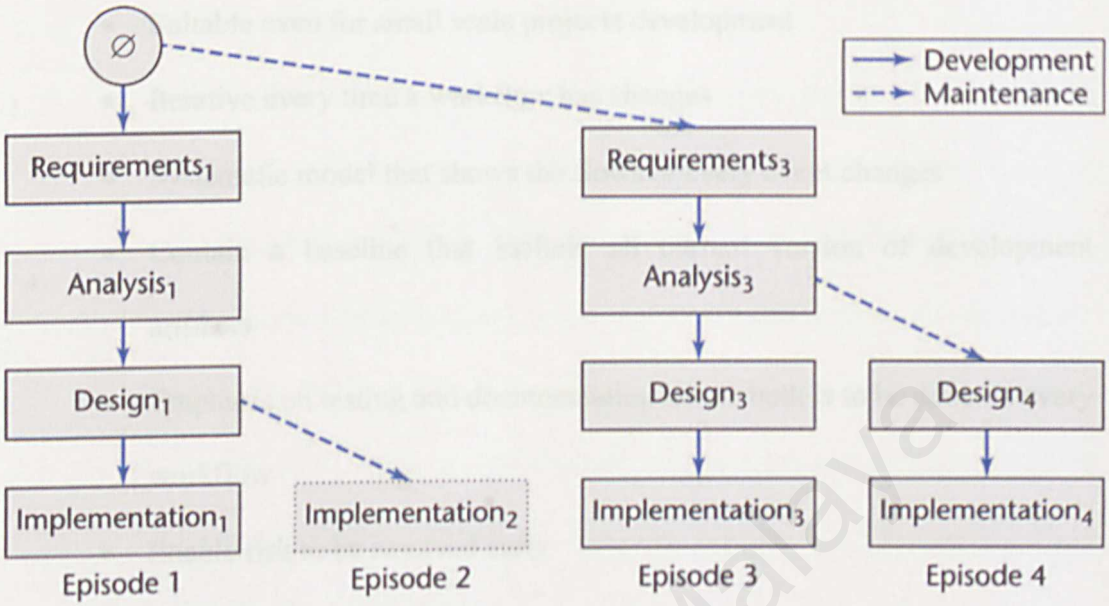



**Figure 3.2: Evolution-Tree life cycle model [24]**

The system was develop from scratch (Ø) followed by the $Requirement_1$, $Analysis_1$, $Design_1$ and $Implementation_1$. When there is something that could not be archive in the $Design_1$, the $Implementation_1$ had to be modified ($Implementation_2$). When a new faster method or algorithm is being used, the requirement had to e change accordingly ($Requirement_3$). There is no documentation and testing in the evolution model as both is carried out through out the development life cycle. The episode of the model increases if there is changes while the development.

### *3.2.1 Strength*

- Suitable even for small scale projects development
- Iterative every time a workflow has changes
- Systematic model that shows the flows of every event changes
- Contain a baseline that include all current version of development artifacts
- Emphasis on testing and documentation where both is to be done on every workflow
- Enable risk to be resolved early

## 3.3 *Information Gathering Methods*

- **Study of existing documentation and project**

  Since this project involve technique that is new and is currently still in research, the main source of requirement is from existing documentation. Internet has been the main source of research since it is borderless throughout the world. Through the internet search engine such as Google and Yahoo!, many research papers and project has been use as case study to understand and develop the system. Existing project and similar thesis paper has also being studied to get the idea of developing the system.

- **Books and references**

  Since the papers and reviews are mostly hard to understand, reference books from library has play an important role of setting up a mind that is ready to analyze the project papers so that it may extract the strength and weakness for each of them and adapt into my project.

- **Discussion**

  Discussions with group members are very important in understanding and clarifying the requirements and the design of the system. Through discussion, views from different angels on the system and development techniques can be obtained. This helped to avoid the blind spots on certain issues due to lack of critical and multi-dimensional analysis.

- **Document Room**

  There a lot of previous thesis from seniors stored at the Document Room of FSKTM. The documents can provide some of the guidelines on how to do the research. These samples are useful to provide basic

guideline and idea on how to generate a good report, by evaluating the strength and weakness of each.

# CHAPTER 4: SYSTEM ANALYSIS

## *4.1 System Analysis*

System analysis is a problem solving technique that decomposes a system into its component pieces for purpose of studying how well those component parts work and interact to accomplish their purpose [23]. System analysis is essential to provide a more though understanding of the project problems and needs that triggered the project. The project domain is studied and analyzed to gain a more understanding of what works, what doesn't and what's needed. The purposes for analysis phase are:

- Gain an understanding of the overall system data flow and systems work.
- Identifying the major components.
- Research on current and latest development technologies.
- Identify the software and hardware requirement to develop and reside the system.
- Analyze and plan control features to develop a robust and reliable system.
- Create a requirement specification definition that include both functional and non-functional requirement.

## *4.2 Requirements Analysis*

In the Leaf Recognition Module, the function that is required is listed below:

### *4.2.1 Functional Requirements*

#### *4.2.1.1 Digital Image Analysis*

| | |
|---|---|
| **Function Name** | Digital Image Analysis |
| **Priority** | Essential |
| **Brief Description** | Allow unknown digital image to be analysis for feature extraction |
| **General Synopsis** | The Digital Image Analysis allows the user to input a query image to the system. The digital image is then processed to acquire the shape, color and structure of the leaf. The features are then used as input for the feature classification process of the neural network. |
| **Features** | 1. Image Analysis<br>Perform analysis process to the images in the image library and mark the tokens.<br>2. Add/Remove species images<br>Allows organization of images for the image library.<br>3. Analysis options<br>Allows the user to configure the threshold and distance of the edge detection process. |

### *4.2.1.2 Neural Network Training*

| | |
|---|---|
| **Function Name** | Neural Network Training |
| **Priority** | Essential |
| **Brief Description** | Allow system to be trained with the image from image library to classify image features |
| **General Synopsis** | The neural network function allows the system to be trained to perform object classification. After the images in the image library being analyzed, the neural network will be trained to recognize the features.. |
| **Features** | 1. Neural Network Toolbox<br>The toolbox allows the user to configure the neural network setting such as training steps, hidden neuron and learning rate. |

### *4.2.1.3 Neural Network Features Classification*

| **Function Name** | Neural Network Features Classification |
|---|---|
| **Priority** | Essential |
| **Brief Description** | Allow system to classify the input image's features |
| **General Synopsis** | The Neural Network Features Classification function allows the system to classify the features extracted from the query image. When the user input an unknown leaf image the image will be analyzed to extract the features required for the image retrieval process. The features will be recognized with a degree of similarity and serves as a query for the image retrieval process. |
| **Features** | 1. Analysis options<br>Allows the user to configure the options such as threshold, distance and weight of the object classification process. |

### *4.2.2 Non-Functional Requirements*

Non-functional requirements are as important as functional requirements. It is defined as constraints under which the system must operate and standard, which must be meet by the delivered system. Two of the most important non-requirements of the Leaf Recognition Module are:

**1) Precision**

The "precision" of the retrieval is the percentage of the K documents that belong to the correct species [14]. PRECISION is the ratio of the number of relevant records retrieved to the total number of irrelevant and relevant records retrieved. It is usually expressed as a percentage.

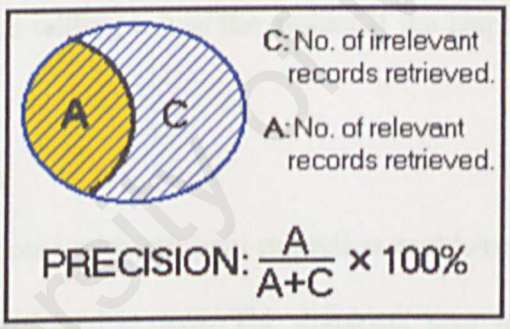



**Figure 4.1: Precision**

**2) Recall**

The "recall" of the retrieval is the percentage of all documents for the correct species that are included in the top K retrieved documents [14] . RECALL is the ratio of the number of relevant records retrieved to the total number of

relevant records in the database. It is usually expressed as a percentage.

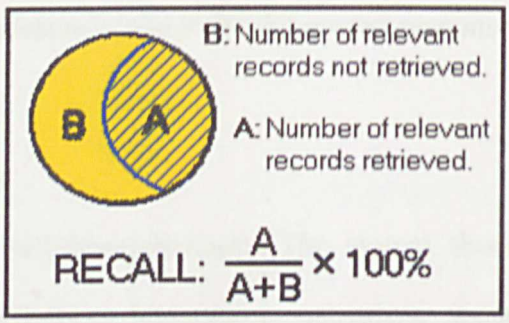



**Figure 4.2: Recall**

On the other hand, the system should be:

3) **Response Time**

The system should able to retrieve information within a reasonable interval time. Users' will be willing to use the system if the response time is within acceptable range.

4) **User–Friendliness**

This system is adopted with minimal migration problems from paper-based system to computer-based system. The graphical user interface should be clear, concise and not cluttered. It is designed in a way that follows the logical flow of the thinking. It displays confirmation messages to assure the user of what has been done and also prompts warning or error messages to help users to correct the mistakes and proceed with the operations. All these are able to guide user better in operating the system.

5) **Reliability**

A system is reliable if it functions as intended by the designer with precision and accuracy and does not produce failures in the typical environment and data integrity is preserved. Thus, this system should be reliable in performing its daily functions and operations. All potential and possible failures and

errors must be taken into account during the design and development stage. Reliability is also responsible to make the correct response and provide error-handing ability.

6) **Understandability**

It is a degree of self-descriptiveness. The system should contain enough information for the users to determine its objectives, assumptions, constraint, input, outputs and status. No excessive information is presented.

7) **Maintainability**

A product is maintainability if the system program are easily modified and tested in the case of updating a process to meet a new requirement, correcting errors or move to a different computer system.

## 4.3 Technology Review

### 4.3.1 Software Development Tools

Due to this proposal is the **Leaf Recognition Module**, the software development tools that takes into considerations are limited to availability of specific functions.

1. *Imaging Functions*

   The main purpose of this module is to develop an engine that was able to recognize digital leaf image. The software development tools used should ready with variety of imaging functions that allows pixel manipulations. It should allow image processing work such as noise removal, edge detection, transformation or segmentation to be done.

2. *High Stability and reliability*

   Due to the involvement of the artificial neural network applications, the software development tools should have high stability. Artificial neural network programming is a high iterative, so the final program may crashes if the software development tools are not stable enough to handle the process. This module also accesses a large image library for the iterative process. The software development tools should be able to access large database with large image storing.

3. *Object-Oriented (OO) Programming and Dynamic Linking*

   Object-oriented programming emphasis on reliable coding by deliberately hiding data and code except through controlled access points. Also, program development is speeded up and made more reliable by OO's inheritance. Reuse of classes through inheritance

make development process much faster and cleverer. Dynamic linking also allows same methods to be carried very different parameters and the runtime system will invoke the right subroutine.

#### *4.3.1.1 Microsoft Visual Basic*

Visual Basic is a fourth-generation language for designing and building applications with graphical user interface (GUI). Visual Basic is one of the most popular and widely used programming languages available today for developing Windows-based, Database and Internet applications.

Windows-based applications are even driven and they need good graphical user interface (GUI) support. Visual Basic can also be used to access database created using database management software (DBMS) such as Access 2000, SQL Server 2000 and FoxPro.

Visual Basic is event-driven; meaning code remains idle until called upon to respond to some event (button pressing, menu selection etc.). An event processor governs Visual Basic. Nothing happens until an event is detected. Once an event is detected, the code corresponding to that event (event procedure) is executed. Program control is then returned to the event processor.

Some Features of Microsoft Visual Basic

- Full set of objects
- Response to mouse and keyboard actions
- Full array of mathematical, string handling, and graphics functions
- Can handle fixed and dynamic variable and control arrays

- Sequential and random access file support
- Useful debugger and error-handling facilities
- Powerful database access tools
- Package & Deployment Wizard makes distributing applications simple

#### *4.3.1.2 Java Technology*

Java is a programming language developed by Sun Microsystems. It's completely object-oriented, flexible than C/C++. You can perform two type of Programming work with java: first is general applications and the second applets. However, applications are not as much developed by using java. Programmers use java to design applets which can be transported via the internet.

In order to develop Java Applets and Applications a java compiler, and runtime environment which is called (Java Development Kit) is needed. Both can be downloaded from Sun's website. The current version is Java 2.0 although most programmers use Java1.3 in common. Java programs can be run in Windows 98, Windows NT, and Solaris 2.3 or higher machines.

According to Sun's definition Java is Simple, Object-oriented, Distributed, Interpreted, Robust, Secure, Architecture neutral, Portable, High performance, Multithreaded, Dynamic language.

| Java vs Visual Basic - A Comparison | | |
|---|---|---|
| **Characteristic** | **Java** | **Visual Basic** |
| **Ease of Development** | **Very good visual development environs, lots of wizards and help** | |

| | | |
|---|---|---|
| No. of syntax elements, routines | 50 commands, 1000s of routines | 300+ commands, 1000s of routines |
| Approachability | +JVM/SDK free & easily available<br>+text editor & browser to run<br>-OO design takes some extra effort<br>-even bigger API to master | +do useful work quite quickly<br>+ VBA is available in many products<br>-Learning edition costs $100<br>-huge API to master |
| Documentation | =mixed quality of written docs<br>+solid electronic reference HTML | -poor written docs; $extra, disorganized<br>-electronic docs $extra; few/poor examples |
| Availability of books, tutorials, courses etc | =lots of books and tutorials<br>=big in college and universities<br>+can do self-training easily | =lots of books and tutorials<br>+MS and other training courses<br>=mixed adoption in colleges/universities |
| **Reliability and stability** | **Constant and rapid change is drawback to both languages** | |
| Major weaknesses | -numerous new classes/APIs<br>-rapidly deprecating classes<br>-huge API, bigger than VB's | -numerous new routines/syntax<br>-just moving to OO technology<br>-always bugs in new features<br>-a mess in data access routines |
| Robustness | +cleaner try/exception handling & event listener models<br>+runtime type, bounds checking<br>+restricted memory manipulation | +automatic syntax checker in editor<br>+direct use of Windows APIs<br>-opened up to direct memory change |
| Security | ++Major advantage is sandlot model | -ActiveX still problematic |
| Overall Reliability | + reasonably reliable at start<br>-bugs creep in early 1.1 SDK<br>-GUI compatibility problems<br>+major ISVs likeOracle, IBM, Sun, Sybase, etc committed to it | +six versions to clear bugs<br>-still islands of instability/bugs<br>-hard to find good advice on what<br>to avoid or how to fix some bugs<br>+the favored tool at Microsoft |
| **Speed of Development** | **Very good native & 3rd party tools** | |
| Editor | -depends on IDE, some are weak | +constantly sets the standard |

| Visual development | =IDEs are quite good | =IDE is quite good |
|---|---|---|
| Debugging/Testing | =client side debugging is good<br>=good but mixed on server debugging<br>=good profiling and test tools | ++best debugger on client by far<br>=improving on server side<br>=some great 3rd party test tools |
| Project size | +medium scale with DBMS<br>+server or large-scale n-tier/web | +small quick and dirty<br>+medium scale with lots of GUI |
| Other | +Advantage is more secure and broad reaching components like JavaBeans, EJBs, servlets, etc.<br>=3rd party tools, libraries, etc. | +Big advantage is huge army of developers<br>=3rd party tools, libraries, etc. |
| **Runtime options** | **Both languages give away 10-30% speed deficit to C/C++** | |
| Cross platform | ++Major advantage | -could be negated if VB emits bytecode |
| Deployment options | =.exes amd bytecode<br>=Beans and EJB components<br>=web servlets<br>+browser applets | =.exes and p-code<br>=ActiveX components<br>=web servlets |
| Speed | +advantage to Java on server<br>+byte code optimizers, trusted compiled apps just getting used | +advantage to VB on client |

**Table 2.2 A Comparison of Java and Visual Basic[30]**

### 4.3.2 *Application Platform*

While the objective of this project is ease the use for botanical user to access a plant knowledge base, this system's application platform should be ease to use and consist of major computer users. Considering the Windows family platform, Window XP has been redesigned to give user a new experience in windows computing. The user will have a more stable and reliable environment than previous version of Windows because it is build based on NT technology and Windows 2000 kernel. With the strengths of Windows 2000 Professional and the best business features of Windows 98, Windows XP Professional is the best desktop operating system.

### *4.3.3 Software Development Tools*

Microsoft Visual Basic .NET (VB.NET) is chosen as the software development tools because VB.NET is the newest implementation of the very popular Visual Basic language and is part of the emerging Microsoft .NET platform. VB.NET includes full object-oriented language features, a new, shared IDE, and many data type changes. VB.NET is the language that provides the easiest transition to the .NET framework for current Visual Basic developers.

Microsoft redesigns Visual Basic using the .NET Framework. OO performance is great for abstractions. Although it required additional operating cost the trade-off is ancestor benefits such as reliable code, reusable functionality.

Visual Basic .NET reflects the following design principles:

- It is recognizably the descendant of Visual Basic. An existing Visual Basic programmer will feel immediately familiar with the language.
- Its syntax and semantics are simple, straightforward, and easy to understand. The language avoids unintuitive features.
- It gives developers the major features of the .NET Framework and is consistent with the framework's conventions.
- It is reasonably upgradeable from Visual Basic.
- Because the .NET Framework explicitly supports multiple computer languages, it works well in a multilanguage environment.
- It is as compatible with previous versions of Visual Basic as possible. Whenever practical, Visual Basic .NET has the same syntax, the same semantics, and the same run-time behavior as its predecessors.

(ms-help://MS.NETFrameworkSDKv1.1/vblsnet/html/vblrfVBSpec1_3.htm)

## 4.4 *Hardware & Software Requirements*

While the system is built under the Microsoft Windows XP platform with the hardware setting listed, the hardware setting will be listed as the optimum requirement. However, it is greatly depends on the image library where the larger the library, the more advance requirement is needed.

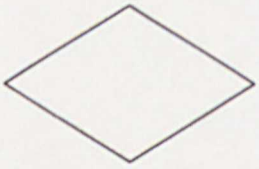
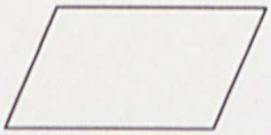
| Optimum Hardware Requirements | Optimum Software Requirements |
|---|---|
| - AMD Athlon™ XP Thoroughbred 1.8 GHz<br>- 768MB RAM<br>- 64MB Graphic Card<br>- Standard input and output<br>- CD-ROM<br>- Others standard computer peripherals | - Microsoft Windows Me / 2000 / XP |

**Table 4.1 Optimum hardware and software requirement**

# CHAPTER 5: SYSTEM DESIGN

## *5.1 Introduction*

System design is defined as those task that focus on the specification of a computer-based solution. Emphasize on the technical or implementation concern of the system. The specification or construction of a technical, computer based solution for the business requirements identified in the system analysis [23]

| Symbol Name | Icon |
|---|---|
| Process |  |
| Decision |  |
| Data |  |
| Document |  |

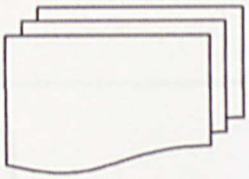
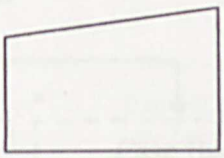
| | |
|---|---|
| Multi-Document |  |
| Manual Input |  |

**Table 5.1: Symbols Flow Chart**

## 5.2 *System Design*

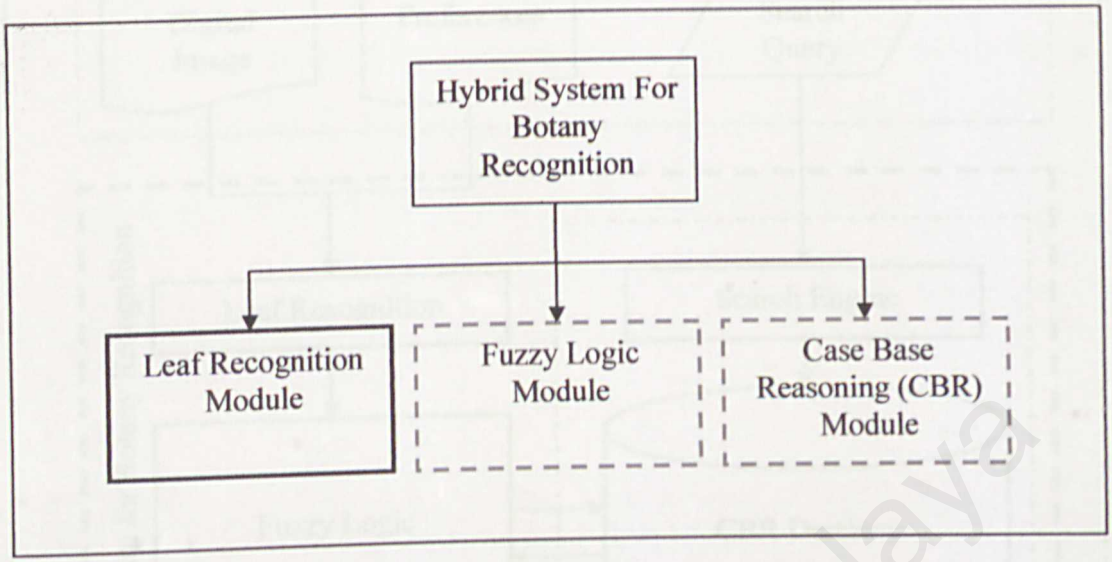



**Figure 5.1: System Hierarchy of Hybrid System for Botany Recognition**

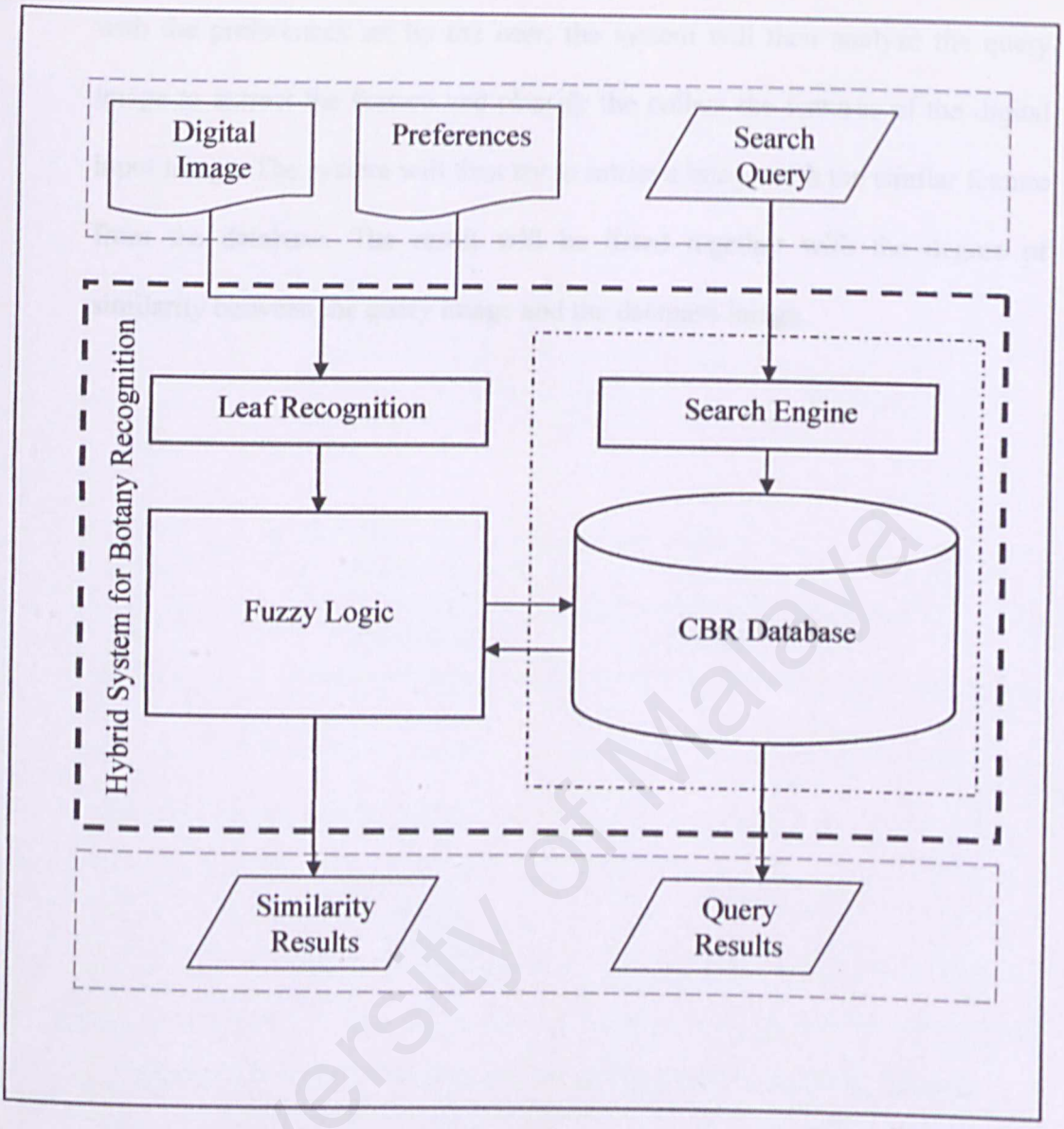



**Figure 5.2: Schematic Description of Hybrid System for Botany Recognition**

Figure 5.1 shows the hierarchy of the propose system. The system consists of three major parts: the **Leaf Recognition Module**, the **Fuzzy Logic Module** and the **CBR Module**. Figure 5.2 shows the interaction between the modules and the external input and output. The user can input the search query as keyword to search the database for related data. The system will attempt to search through the database for related keyword and retrieve the list of query result to be review. The user can also input a digital image as a query. Together

with the preferences set by the user, the system will then analyze the query image to extract the feature and classify the collect the features of the digital input image. The system will then try to retrieve image with the similar feature from the database. The result will be listed together with the degree of similarity between the query image and the database image.

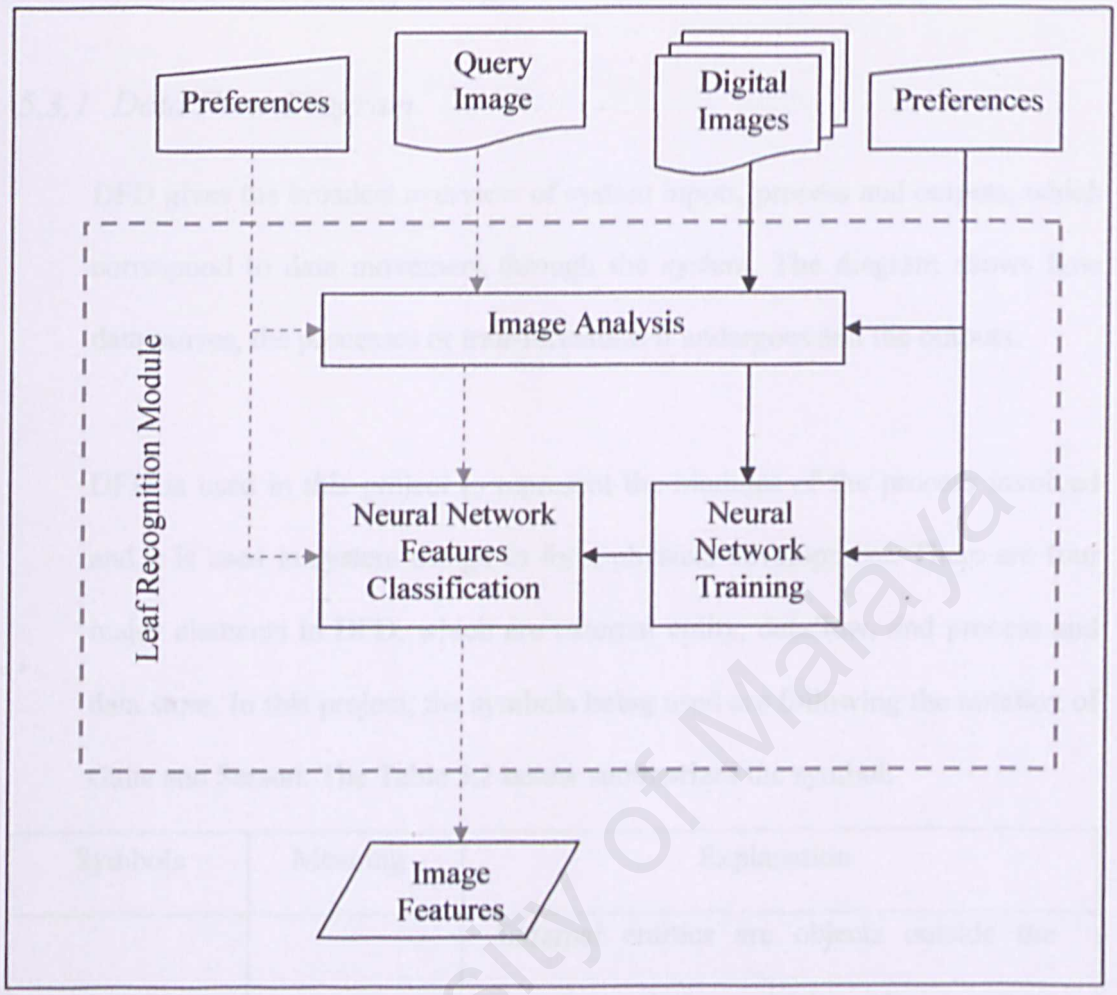



**Figure 5.3: Schematic Description of the Leaf Recognition Module**

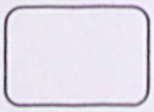
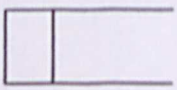
From figure 5.3, a collection of digital image is served as an input for the system. The images will be categorized in its corresponding features. The structured image collection will then being analyze and trained by the neural network so the neural network will recognize the features available in the image collection. After the training, the neural network will act as a feature classification engine for the query image. When the user input a query image, the system will attempt to extract the features available

## 5.3 *System Functionality Design*

### 5.3.1 *Data Flow Diagram*

DFD gives the broadest overview of system inputs, process and outputs, which correspond to data movement through the system. The diagram shows how data moves, the processes or transformation it undergoes and the outputs.

DFD is used in this project to represent the modules of the process involved and it is used in system design to form physical development. There are four major elements in DFD, which are external entity, dataflow, and process and data store. In this project, the symbols being used are following the notation of Gane and Sarson. The Table 5.2 below summarizes the symbol.

| Symbols | Meaning | Explanation |
|---|---|---|
| External Entity | External Entity | External entities are objects outside the system, with which the system communicates. External entities are sources and destinations of the system's inputs and outputs. |
|  | Process | A process transforms incoming data flow into outgoing data flow. |
|  | Data Store | Data stores are repositories of data in the system. They are sometimes also referred to as files. |

| | | |
|---|---|---|
| → | Data Flow | Dataflow are pipelines through which packets of information flow. Label the arrows with the name of the data that moves through it. |

**Table 5.2: DFD Symbols**

### *5.3.2 Context Diagram*

A context diagram is a top level (also known as Level 0) data flow diagram. It only contains one process node (process 0) that generalizes the function of the entire system in relationship to external entities.

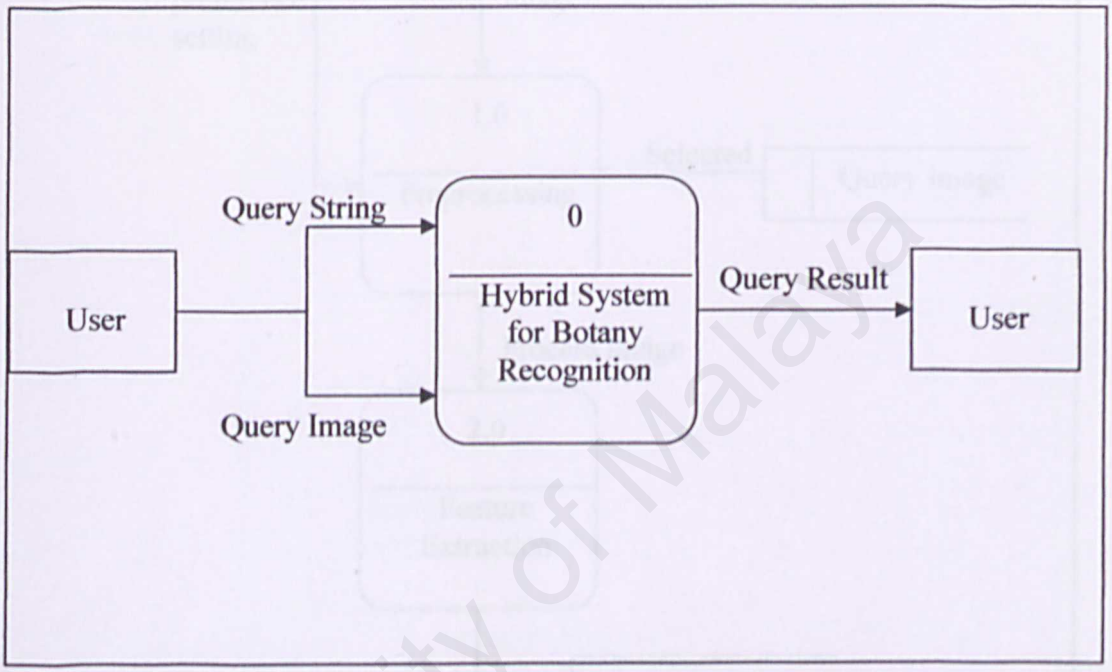



**Figure 5.4: Context Diagram of the Leaf Recognition Module**

### 5.3.3 *Digital Image Analysis*

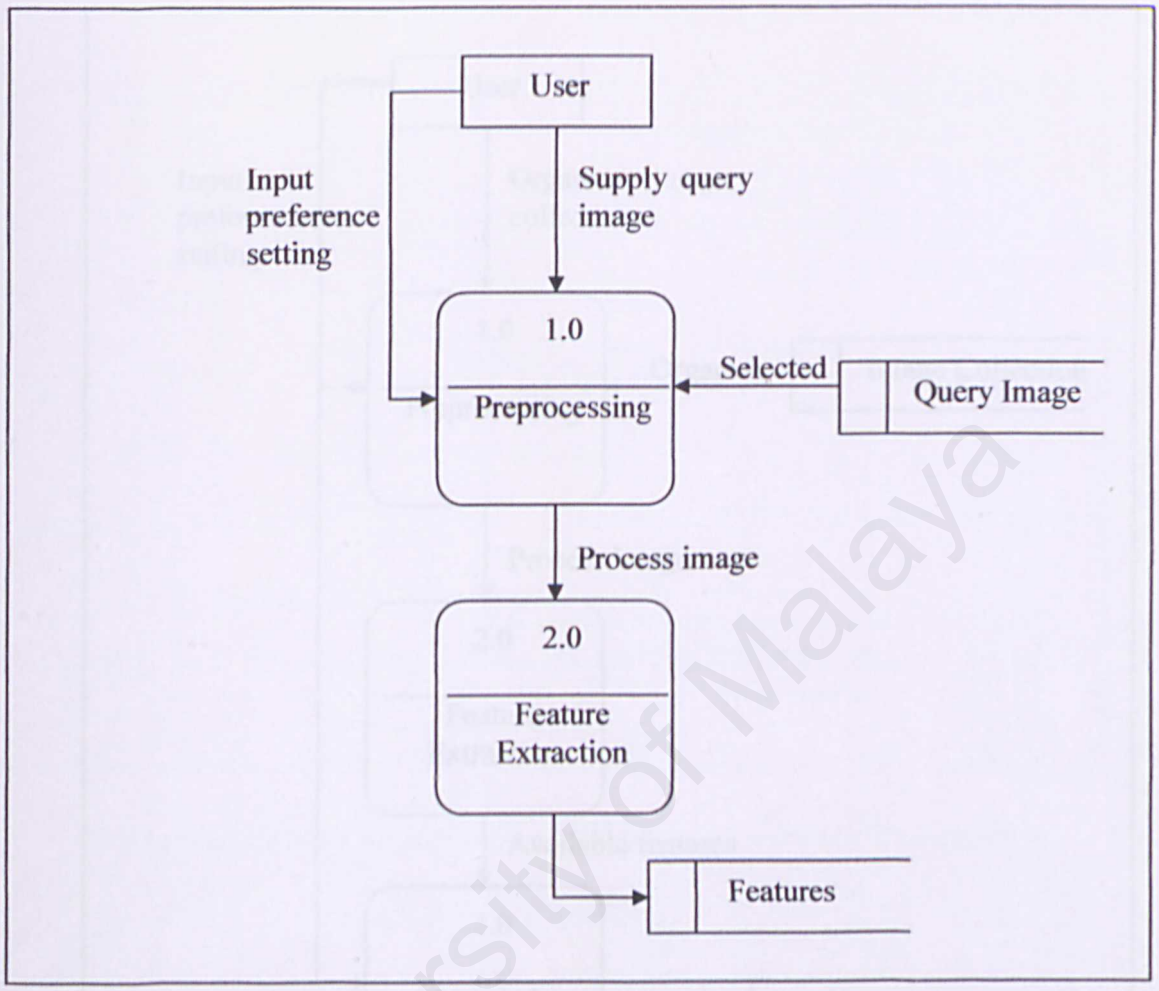



**Figure 5.5: Data Flow Diagram of the Digital Image Analysis Function**

From Figure 5.5, the users first supply the query image in supported digital image format. The query image will then go through a preprocessing process where process such as segmentation, noise removal or conversion is done to the query image. The processed query image is then used to extract out the features for the use in either the neural network training or classification function.

### 5.3.4 *Neural Network Training*

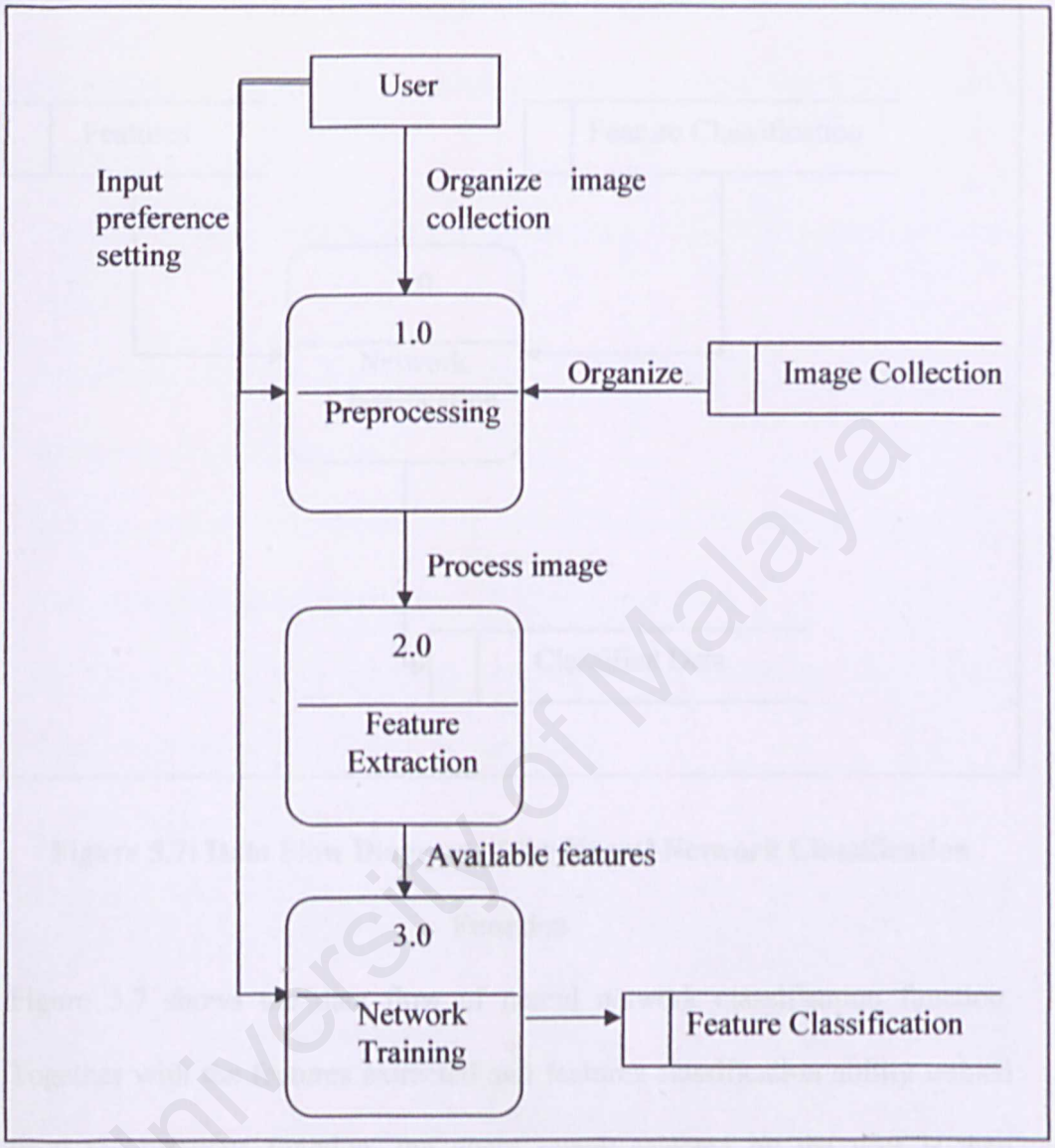



**Figure 5.6: Data Flow Diagram of the Neural Network Training Function**

Figure 5.6 shows the data flow of neural network training function. The user first organizes the available image into similar feature collection. Then the organized image will serve as training input for the system to recognize the features. Together with the preference set, the system is trained to recognize the features with the minimum error.

### 5.3.5 *Neural Network Features Classification*

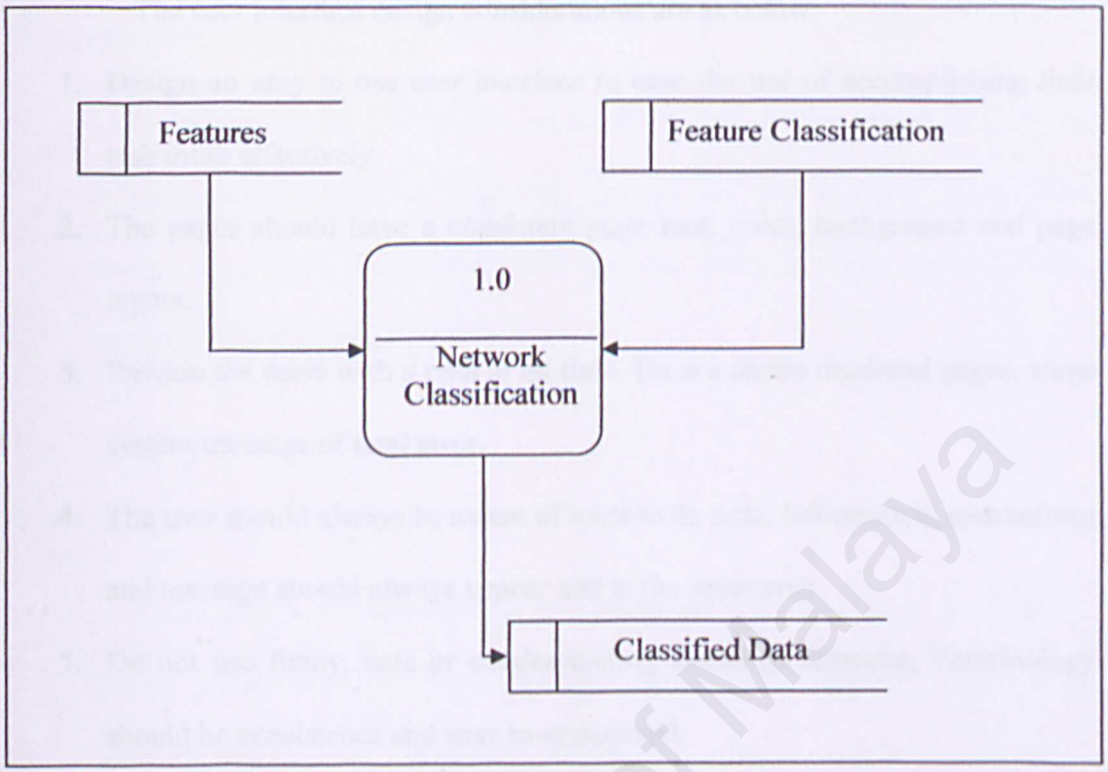



**Figure 5.7: Data Flow Diagram of the Neural Network Classification Function**

Figure 5.7 shows the data flow of neural network classification function. Together with the features extracted and features classification ability trained from the previous function, the query image selected by the user is then classify its features in to known features categories. The result is shown with a degree of similarity between the features.

## 5.4 *User Interface Design*

The user interface design considerations are as below:

1. Design an easy to use user interface to ease the use of accomplishing their task more effectively.
2. The pages should have a consistent page font, color, background and page layout.
3. Provide the users with a path at all time. Do not create dead-end pages, show system message of fatal error.
4. The user should always be aware of what to do next. Information, instructions and message should always appear and in the same area.
5. Do not use funny, cute or condescending words or sentence. Terminology should be consistence and easy to understand.

The sample screen shot is as below:

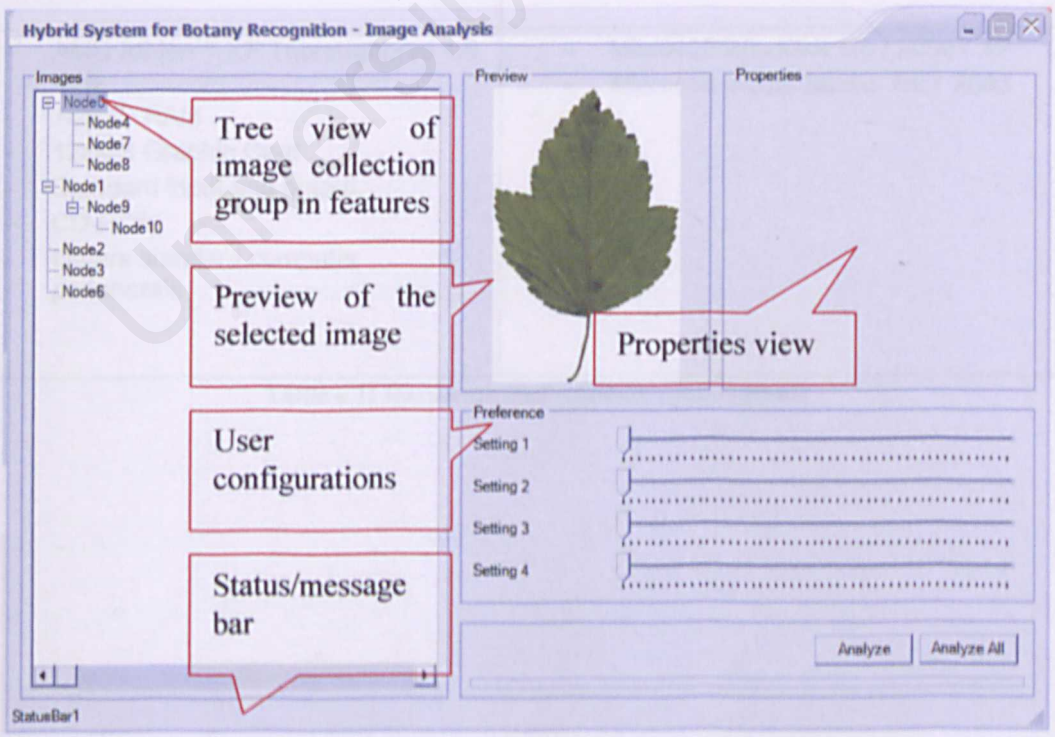



**Figure 5.8: Sample screen shot for image analysis**

# CHAPTER 6: SYSTEM IMPLEMENTATION

## 6.1 *Introduction*

In this phase, the system requirements and design model of a system will be converted into a workable product. In this section, development environment will be stated and coding methodology and algorithm implementation will be discussed.

## 6.2 *Development Environment*

Using the suitable hardware and software not only help to speed up the system development but also determine the success of the project. The hardware and software tools used to develop the entire system are as below:

| Hardware Requirements | Software Requirements |
|---|---|
| - AMD Athlon™ XP Thoroughbred 1.8 GHz<br>- 768MB RAM<br>- 128MB Graphic Card<br>- Standard input and output<br>- CD-ROM<br>- Others standard computer peripherals | - Microsoft Windows Me / 2000 / XP<br>- Microsoft Visual Studio .NET 2003 |

**Table 6.1: Hardware and software requirements**

## 6.3 *Coding Methodology*

4th generation language has been chosen as a development tools in this system because of its simplicity and maintainability. Good programming style will also be implemented such as using consistence and meaningful variable names, code layout for readability and coding design.

The system will be implemented using top-down approach. High level module will be implemented first then followed by the lower level module.

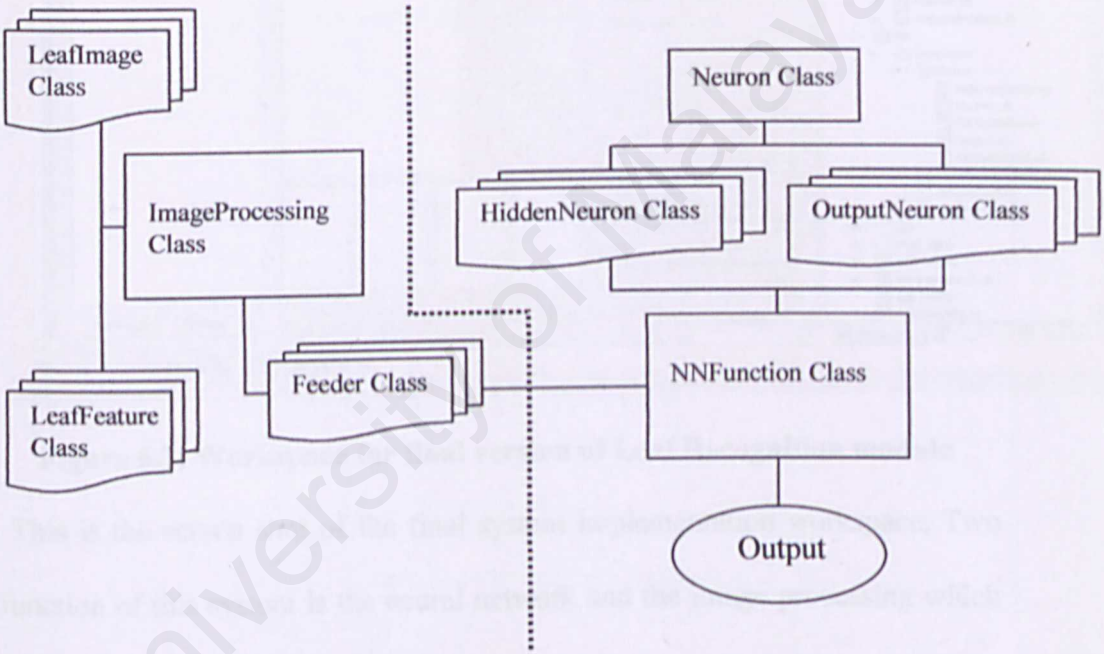



**Figure 6.1: System coding design for Leaf Recognition module**

## 6.4 FINAL SYSTEM IMPLEMENTATION


**Figure 6.2: Workspace for final version of Leaf Recognition module**

This is the screen shot of the final system implementation workspace. Two major function of this system is the neural network and the image processing which are coded first.

The image processing component implements Prewitt edge detection algorithm which has better result than the Robert's or Sobel's yet implementation is not as complicated as the Canny's. This component also implements a thinning algorithm proposed by R. Fisher, S. Perkins, A. Walker and E. Wolfart[32]. The image processing (IP) component is divided into several classes.

- ImageProcessing Class

  This class contains the major functions of processing a digital image before it is used in the Leaf Recognition. The classes contain the edge

detection algorithm, which implement the Prewitt algorithm. Prewitt edge detection uses two convolution masks $P_1$ and $P_2$ which are as follow:

$$P_1 = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \quad P_2 = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

$$\text{Edge Magnitude} = \sqrt{P_1^2 + P_2^2}$$

**Figure 6.3: Prewitt convolution mask and edge magnitude**

The converted algorithm is as follows:

```
'  Function of Prewitt Edge Detection using function PrewittConv2()
    Private Function EdgeDetect(ByVal _imgLeaf As LeafImage)
        '
        Dim int_x As Integer
        Dim int_y As Integer
        Dim resolution As New Rectangle(0, 0, Me.intWidth, Me.intHeight)
        '
        Me.imgEdge = _imgLeaf.Bitmap.Clone()
        Me.BitmapClear(imgEdge)
        '
        For int_y = Me.intHeight - 3 To 2 Step -1
            For int_x = Me.Width - 3 To 2 Step -1
                Me.imgEdge.SetPixel(int_x, int_y, _
                    Me.PrewittConv2(int_x, int_y, _imgLeaf.Bitmap))
            Next int_x
        Next int_y
        '
        _imgLeaf.bmpAfterED = Me.imgEdge.Clone
        '  Loop through the bitmap and filter pixels with threshold
        '  and thinning process
        Me.dblThreshold = (Me.dblThreshold * Me.dblMaxPixelValue)
        '
        For int_y = 0 To Me.intHeight - 1
            For int_x = 0 To Me.Width - 1
                Me.imgEdge.SetPixel(int_x, int_y, _
                    Me.FilterThreshold(Me.imgEdge.GetPixel(int_x, int_y)))
            Next int_x
        Next int_y
        _imgLeaf.bmpAfterThreshold = Me.imgEdge.Clone
        '
    End Function 'EdgeDetect
```

```
'
    '  For the edge detection the "Prewitt" Alogrithm is used:
    '    intConvY        intConvX
    '   (1 0 -1)        (-1 -1 -1)
    '   (1 0 -1)        ( 0  0 0)
    '   (1 0 -1)        ( 1  1  1)
    '  Function return a grayscale value and save to buffer image (imgedge)
    Private Function PrewittConv2(ByVal _intx As Integer, _
        ByVal _inty As Integer, ByVal _imgLeaf As Bitmap) As Color
        '
        Dim dblConvX As Double
        Dim dblConvY As Double
        Dim dblMagnitude As Double
        '
        '  Convolution of the 2 masks
        dblConvX = (_imgLeaf.GetPixel(_intx - 1, _inty + 1).GetBrightness + _
                _imgLeaf.GetPixel(_intx, _inty + 1).GetBrightness + _
                _imgLeaf.GetPixel(_intx + 1, _inty + 1).GetBrightness) - _
```

```
            (_imgLeaf.GetPixel(_intx - 1, _inty - 1).GetBrightness + _
            _imgLeaf.GetPixel(_intx, _inty - 1).GetBrightness + _
            _imgLeaf.GetPixel(_intx + 1, _inty - 1).GetBrightness)
        dblConvY = (_imgLeaf.GetPixel(_intx - 1, _inty + 1).GetBrightness + _
            _imgLeaf.GetPixel(_intx - 1, _inty).GetBrightness + _
            _imgLeaf.GetPixel(_intx - 1, _inty - 1).GetBrightness) - _
            (_imgLeaf.GetPixel(_intx + 1, _inty + 1).GetBrightness + _
            _imgLeaf.GetPixel(_intx + 1, _inty).GetBrightness + _
            _imgLeaf.GetPixel(_intx + 1, _inty - 1).GetBrightness)
        '
        dblMagnitude = Math.Sqrt(Math.Pow(dblConvX, 2) + _
            Math.Pow(dblConvY, 2)) * 100 / 3
        '
        ' To get the max pixel brightness
        If dblMagnitude > Me.dblMaxPixelValue Then
            Me.dblMaxPixelValue = dblMagnitude
        End If

        ' To get the min pixel brightness
        If dblMagnitude < Me.dblMinPixelValue Then
            Me.dblMinPixelValue = dblMagnitude
        End If
        '
        Return Me.GrayScaleColor(dblMagnitude)
        '
    End Function 'Convolution
```

**Table 6.2: Sample coding for edge detection**

The ImageProcessing Class also contains a thinning algorithm which is used to reduce the width of the resulted edge pixels into one pixel width. This algorithm mainly is to ease the feeder collecting algorithm.

| | | |
|---|---|---|
| 0 | 0 | 0 |
| | 1 | |
| 1 | 1 | 1 |

| | | |
|---|---|---|
| | 0 | 0 |
| 1 | 1 | 0 |
| | 1 | |

**Figure 6.4: Convolution mask for morphological thinning**

This process is iterated several times. At each iteration, the image is first convolved with the left mask then the right. Then the image is convolved with the remaining six *90°* rotations of the two elements.

1)

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |
| 1 | 1 | 1 |

2)

| | | |
|---|---|---|
| | 1 | |
| 1 | 1 | 1 |
| | 1 | |

3a)

| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | | |

3b)

| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| | | 0 |

**Figure 6.5: Some applications of thinning**

Then the resulted image is convolved with the mask in figure 6.5. Mask figure 6.5.1 and figure 6.5.2 simply finds the boundary of a binary object, *i.e.* it deletes any foreground points that don't have at least one neighboring background point. Mask figure 6.5.3 is used to remove undesirable short spurs. These mask is also performed several iteration and each iteration each element must be used in each of its four *90°* rotations.

The algorithm in VB.NET is as follows:

```
'
    ' Function of thinning algorithm 2 that finds pattern in function Pattern() 1 to 8
    ' and remove unused pixel to produce a one pixel width edge
    Private Sub Thinning2(ByVal _imgLeaf As LeafImage)
        Dim width As Integer
        Dim height As Integer
        Dim n As Integer
        '
        For i As Integer = 1 To 5
            For n = 1 To 8 Step 1
                For width = 0 To Me.intWidth - 1
                    For height = 0 To Me.intHeight - 1
                        If Me.Pattern(width, height, n) Then
                            Me.imgEdge.SetPixel(width, height, Me.BACKGROUNDCOLOR)
                        End If
                    Next height
                Next width
            Next n
        Next i
        '
        _imgLeaf.bmpAfterThin = Me.imgEdge.Clone
        '
    End Sub 'Thinning2
```

```
'
    ' Function of pruning algorithm that finds unused edge after the thinning algorithm
    ' and remove it by comparing the pattern in function Pattern() 9 - 17
    Private Sub Pruning(ByVal _imgLeaf As LeafImage)
        Dim width As Integer
        Dim height As Integer
        Dim n As Integer
        '
        For i As Integer = 1 To 2
            For n = 9 To 17 Step 1
                For width = 0 To Me.intWidth - 1
                    For height = 0 To Me.intHeight - 1
                        If Me.Pattern(width, height, n) Then
                            Me.imgEdge.SetPixel(width, height, Me.BACKGROUNDCOLOR)
                        End If
                    Next height
                Next width
            Next n
        Next i
        '
        _imgLeaf.bmpAfterThin = Me.imgEdge.Clone
        '
    End Sub 'Pruning
```

**Table 6.3: Sample coding for thinning algorithm**

- LeafImage Class

  LeafImage class contains all information for a leaf image inserted to the system. It contains the file name, file path, feature, feeder and tokens of this image used for training neural network.

- LeafFeature Class

  This class contains information of a feature used to categorize similar image. It contains the feature name and feature code of a category.

- Feeder Class

  This class is used as part of the LeafImage class that stored the points on the image that will be used to collect the tokens for neural network training. Each feeder is a set of two coordinate points on the image. Each token is a set of cosinus and sinus value of the feeder.

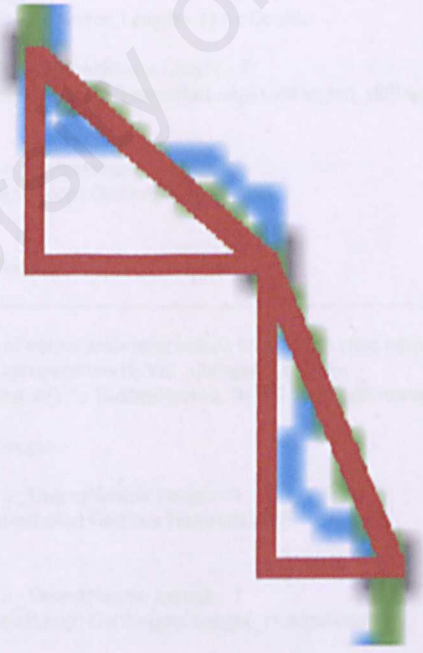


**Figure 6.6: Example of a feeder point**

The other main module is the neural network (NN) module. The neural network architecture chosen to be used in this system is the backpropagation neural network. This neural network has the ability to achieve a balance

between responding correctly to the input that is used in the training (memorization) and give reasonable response to input that is similar but not identical to the input used in the training (generalization). This module is also divided into several components:

- NNFunctions Class

  As this Leaf Recognition module uses the backpropagation neural network, the NNFunctions class contains the major functions for this neural network such as the activation function, feedforward process, backpropagation process and calculation of total squared error function. The sample coding is as follows:

```
'
    ' Feedforward process of neural network to calculate output signal
    Public Function FeedForward(ByVal _dblInput As Array, _
      ByVal _HiddenNeuron() As HiddenNeuron, ByVal _OutputNeuron() As OutputNeuron)
      '
      Dim intLoop As Integer
      Dim dblZ_out(_HiddenNeuron.Length - 1) As Double
      '
      For intLoop = 0 To _HiddenNeuron.Length - 1
        dblZ_out.SetValue(_HiddenNeuron(intLoop).GetOutput(_dblInput), intLoop)
      Next intLoop
      '
      For intLoop = 0 To _OutputNeuron.Length - 1
        _OutputNeuron(intLoop).GetOutput(dblZ_out)
      Next intLoop
      '
    End Function 'FeedForward
```

```
'
    ' Feedback process of output error information to calculate error term
    Public Function Backpropagation(ByVal _dblInput As Array, _
      ByVal _HiddenNeuron() As HiddenNeuron, ByVal _OutputNeuron() As OutputNeuron)
      '
      Dim intLoop As Integer
      '
      For intLoop = 0 To _OutputNeuron.Length - 1
        _OutputNeuron(intLoop).GetErrorTerm(intLoop)
      Next intLoop
      '
      For intLoop = 0 To _OutputNeuron.Length - 1
        _OutputNeuron(intLoop).GetWeightChange(_HiddenNeuron)
      Next intLoop
      '
      '
      For intLoop = 0 To _HiddenNeuron.Length - 1
        _HiddenNeuron(intLoop).GetErrorTerm(_OutputNeuron, intLoop)
      Next intLoop
      '
      For intLoop = 0 To _HiddenNeuron.Length - 1
        _HiddenNeuron(intLoop).GetWeightChange(_dblInput)
      Next intLoop
      '
    End Function 'Backpropagation
```

**Table 6.4: Sample coding for feedforward and backpropagation**

- Neuron Class

This class is the base class for HiddenNeuron and OutputNeuron Class. It contains the basic functions perform to a neuron such as weight initialization and calculation of weighted input signal. For a better learning result, this neuron uses the Nguyen-Widrow initialization.

```
'
'  Training initialization of weight value using NGUYEN-WIDROW initialization
Public Overridable Function InitializeWeight(ByVal _InputCount As Integer, _
  ByVal _HiddenCount As Integer)
  Try
    Dim intLoop As Integer
    Dim scaleFac As Double = 0.7 * (Math.Pow(_HiddenCount, (1 / _InputCount)))
    Dim dblWeightOld As Double = 0
    '
    Me.dblWeight.Capacity = _InputCount
    '
    For intLoop = 0 To _InputCount - 1
      Me.dblWeight.Add(Me.GetRandNum(Me.clsGetOpt.Scale))
    Next intLoop
    '
    For intLoop = 0 To _InputCount - 1
      dblWeightOld = dblWeightOld + (Math.Pow(Me.dblWeight(intLoop), 2))
    Next intLoop
    '
    For intLoop = 0 To _InputCount - 1
      Me.dblWeight(intLoop) = (Me.dblWeight(intLoop) * scaleFac) / _
          (Math.Sqrt(dblWeightOld))
    Next intLoop
    '
    Me.dblBiasWeight = Me.GetRandNum(Me.clsGetOpt.Scale)
    '
  Catch ex As Exception
    '
  End Try
End Function 'InitializeWeight
```

**Table 6.5: Sample coding for weight initialization**

- HiddenNeuron Class

HiddenNeuron class provides additional function as a hidden neuron of a multilayer backpropagation neural network. This class inherits neuron base class so it has all neuron functions.

- OutputNeuron Class

OutputNeuron class provides additional function as a output neuron of a neural network. This class inherits neuron base class so it has all neuron functions.

# CHAPTER 7: SYSTEM TESTING

## *7.1 Introduction*

In this chapter the developed system will be tested using a set of leaf image downloaded from the internet. In the image processing module, unit testing will be conducted to check every function of the module performed optimally. Then the neural network will be tested for it memorization and its generalization. Figure 7.1 show list of images used in this testing phase.

Besides testing of the functionality of the system, the purpose of this testing phase also allows the system to have a default optimum configuration to be used in most of the cases. Both the training and recognition has the configuration for user to input the settings. This testing phase will find out the optimum configuration value to be set as a default.


**Figure 7.1: Testing image list**

## 7.2 Module Testing

### 7.2.1 Image Processing Module

#### 7.2.1.1 Edge Detection

Image Species1_01.jpg has been used as testing subject. Figure 7.2 shows the result of using threshold value 0.5.

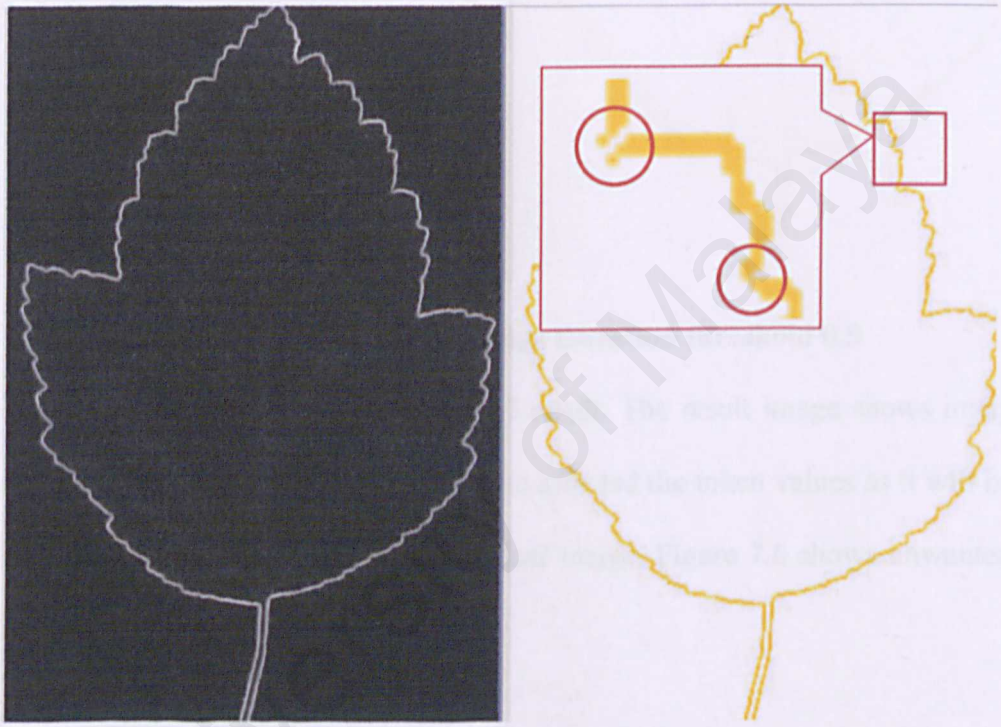


**Figure 7.2: Edge detection using threshold 0.5**

When using the threshold value 0.5, note that there are points that is 'disconnected' from each other. Figure 7.3 shows thinning algorithm perform on the resulted image. There are lost edge starting from the disconnected pixel.

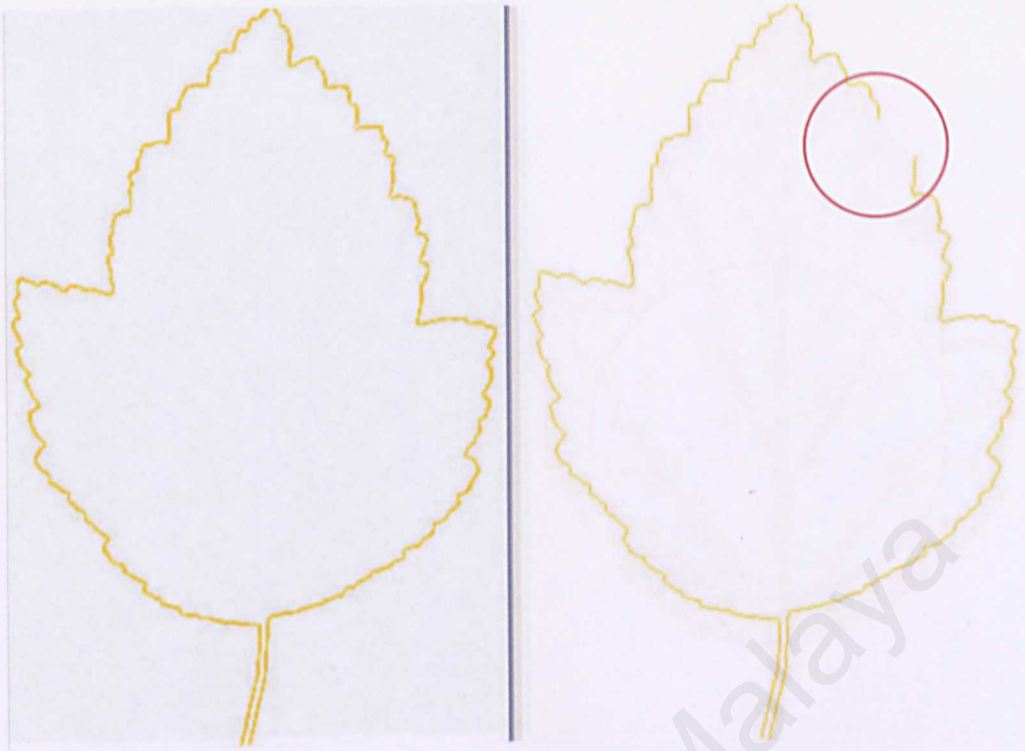


**Figure 7.3: Thinning on edge detection threshold 0.5**

Figure 7.4 show a threshold value 0.05 result. The result image shows many unwanted edge is detected. This will also affected the token values as it will be considered as part of the shape of the leaf image. Figure 7.6 shows unwanted edge is taken as feeder points.

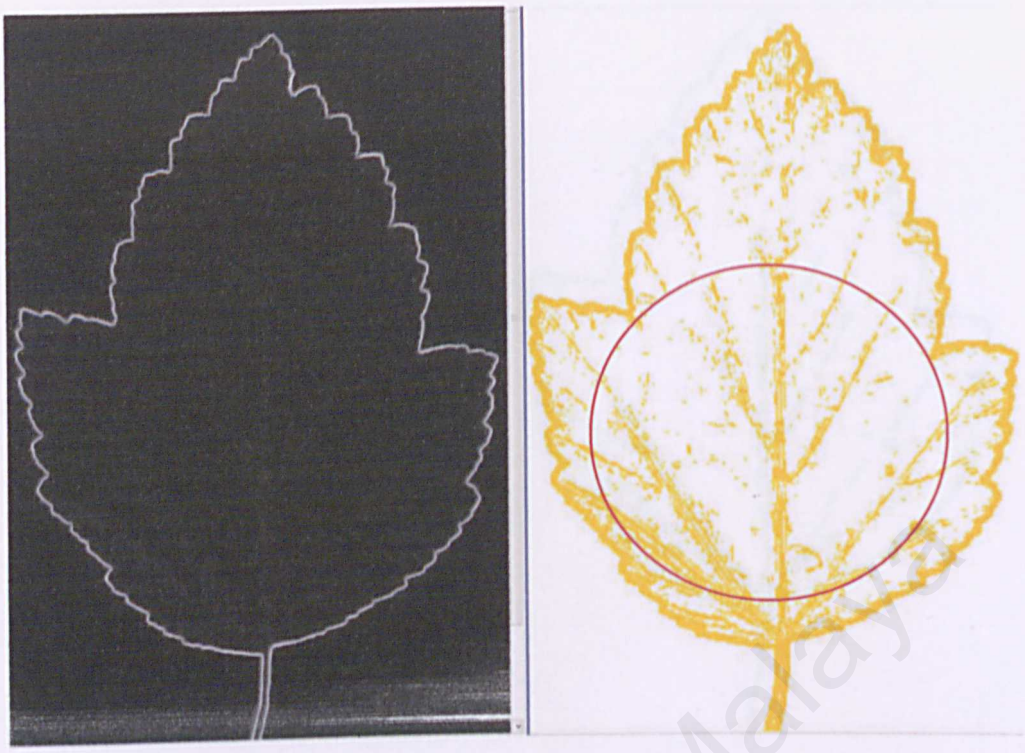


**Figure 7.4: Edge detection using threshold 0.05**

**Figure 7.5: Thinning on edge detection threshold 0.05**

**Figure 7.6: Feeding on edge detection threshold 0.05**

After several trying on different images, optimal edge detection threshold values falls between 0.25 to 0.3. Figure 7.7, 7.8, 7.9 shows the result of threshold value 0.3.

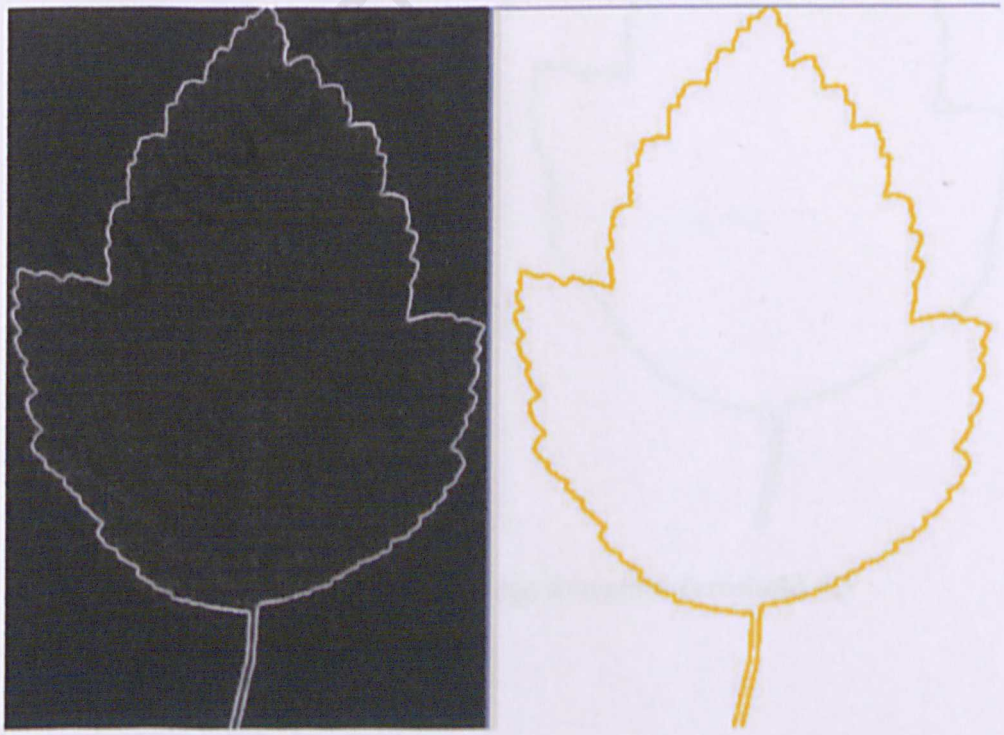


**Figure 7.7: Edge detection using threshold 0.3**

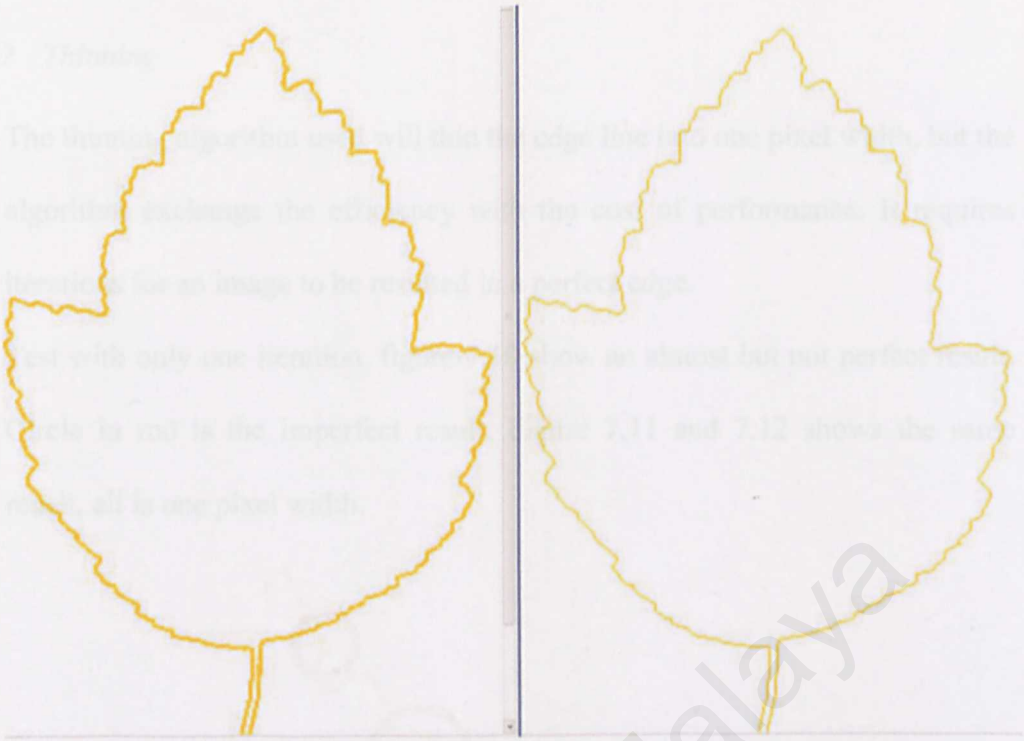


**Figure 7.8: Thinning on edge detection threshold 0.3**

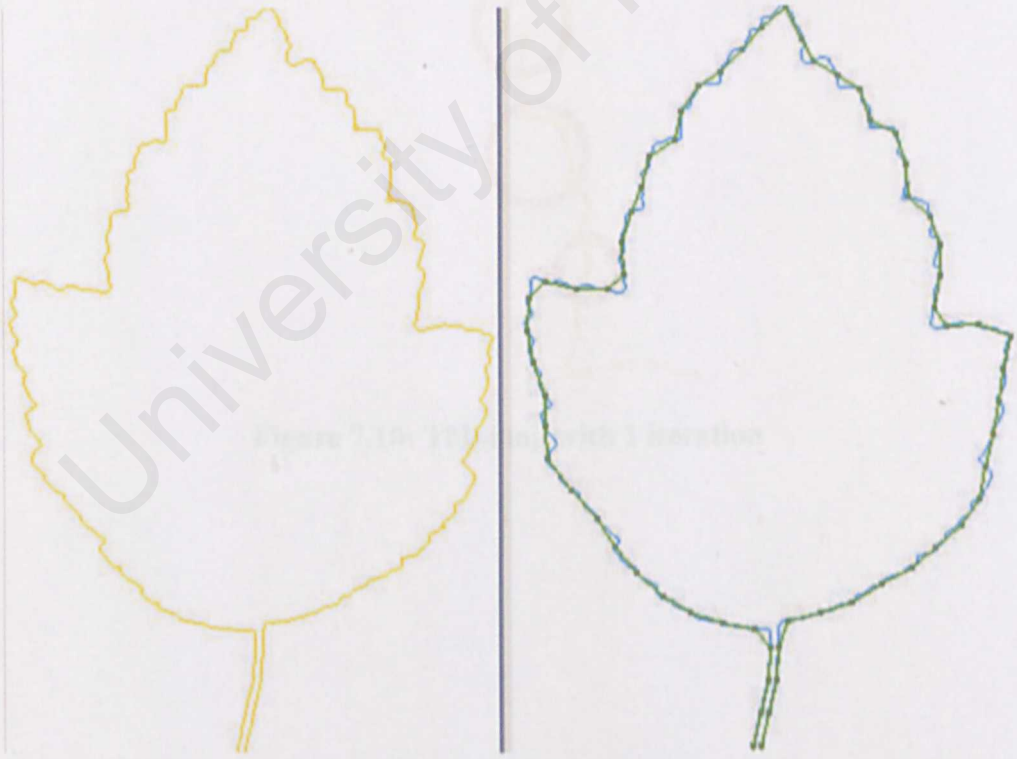


**Figure 7.9: Feeding on edge detection threshold 0.3**

*7.2.1.2 Thinning*

The thinning algorithm used will thin the edge line into one pixel width, but the algorithm exchange the efficiency with the cost of performance. It requires iterations for an image to be resulted in a perfect edge.

Test with only one iteration, figure 7.10 show an almost but not perfect result. Circle in red is the imperfect result. Figure 7.11 and 7.12 shows the same result, all is one pixel width.

**Figure 7.10: Thinning with 1 iteration**

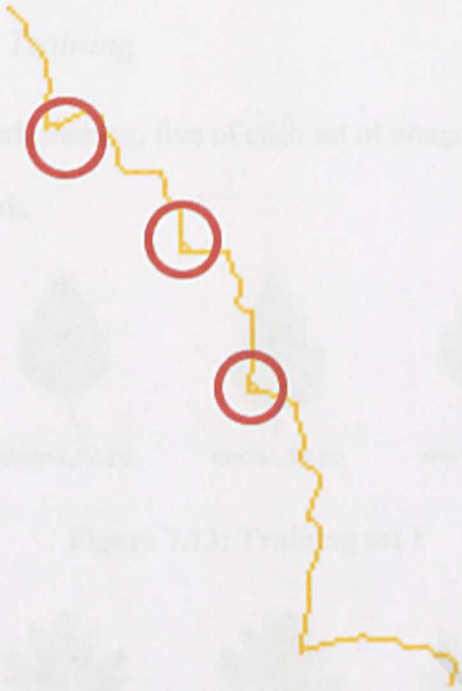


**Figure 7.11: Thinning with 2 iteration**

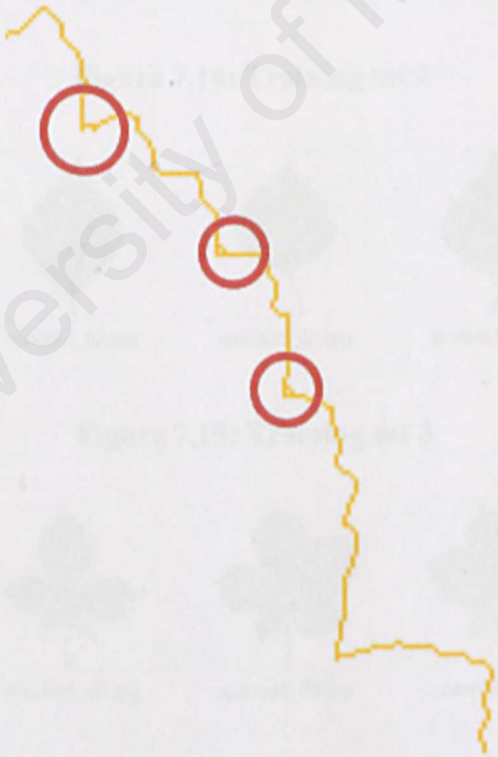


**Figure 7.12: Thinning with 3 iteration**

### 7.2.2 *Neural Network Training*

For the neural network training, five of each set of image is used as training set for the neural network.

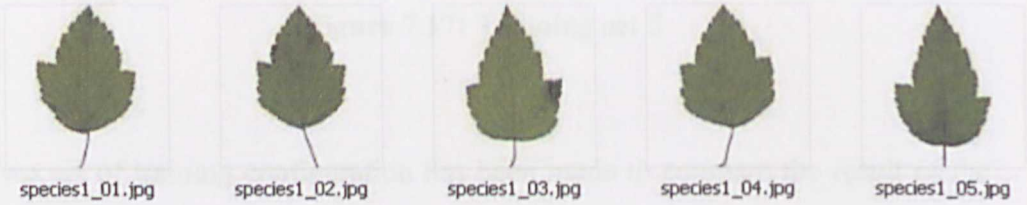



**Figure 7.13: Training set 1**


**Figure 7.14: Training set 2**

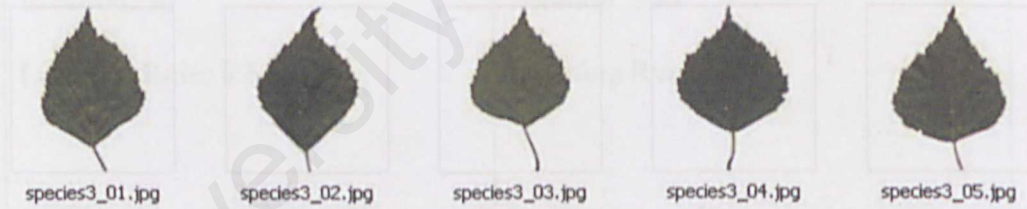



**Figure 7.15: Training set 3**

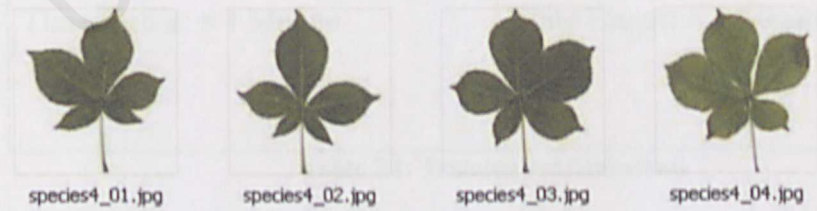



**Figure 7.16: Training set 4**


**Figure 7.17: Training set 5**

Two set of training configuration has been made to compare the result of the memorization and generalization. One is 50 iterations and one is 500 iterations.

| Training 1: | Training 2: |
|---|---|
| Input Count: 202 | Input Count: 202 |
| Hidden Neuron Count: 20 | Hidden Neuron Count: 20 |
| Output Neuron Count: 5 | Output Neuron Count: 5 |
| Iteration: 50 | Iteration: 500 |
| Learning Rule: 0.3 | Learning Rule: 0.3 |
| Total Squared Error: 0.004 | Total Squared Error: $1.461 \times 10^{-12}$ |
| Time Elapse: < 1 Minute | Time Elapse: < 4 Minute |

**Table 7.1: Training configurations**

After 50 iterations, the total squared error of the training result is 0.004, while after 500 iterations the total squared error is $1.461 \times 10^{-12}$.

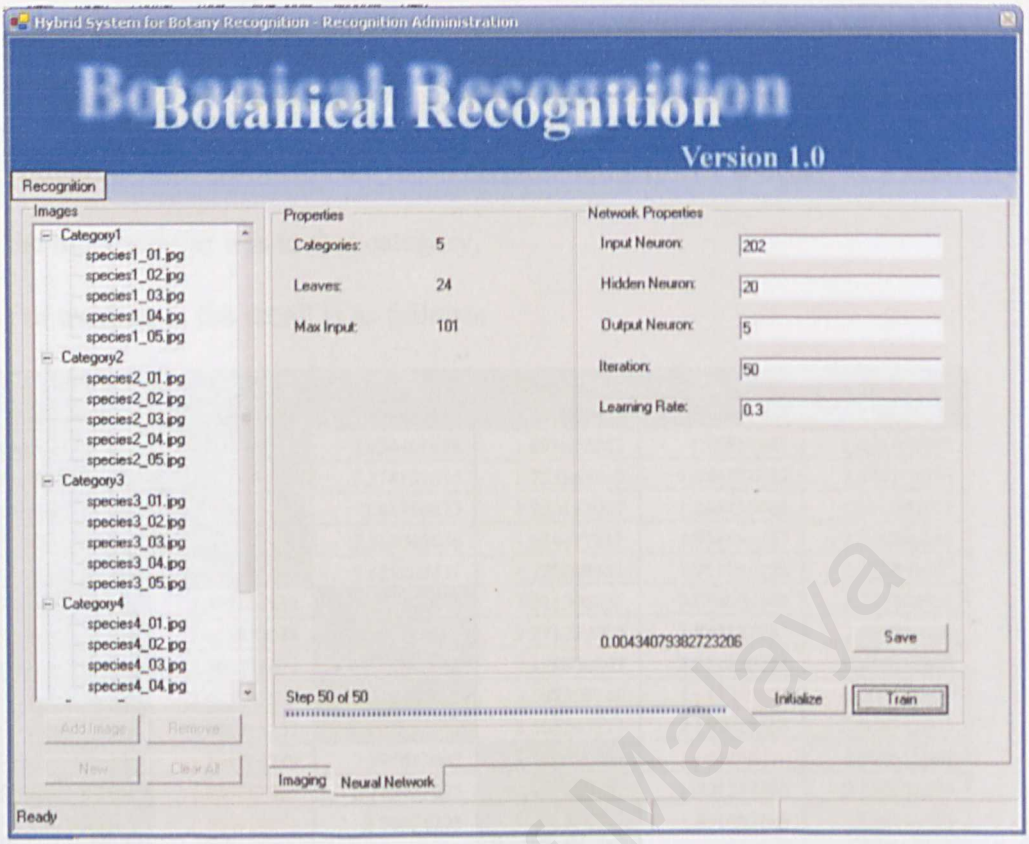



**Figure 7.18: Result of training 1**

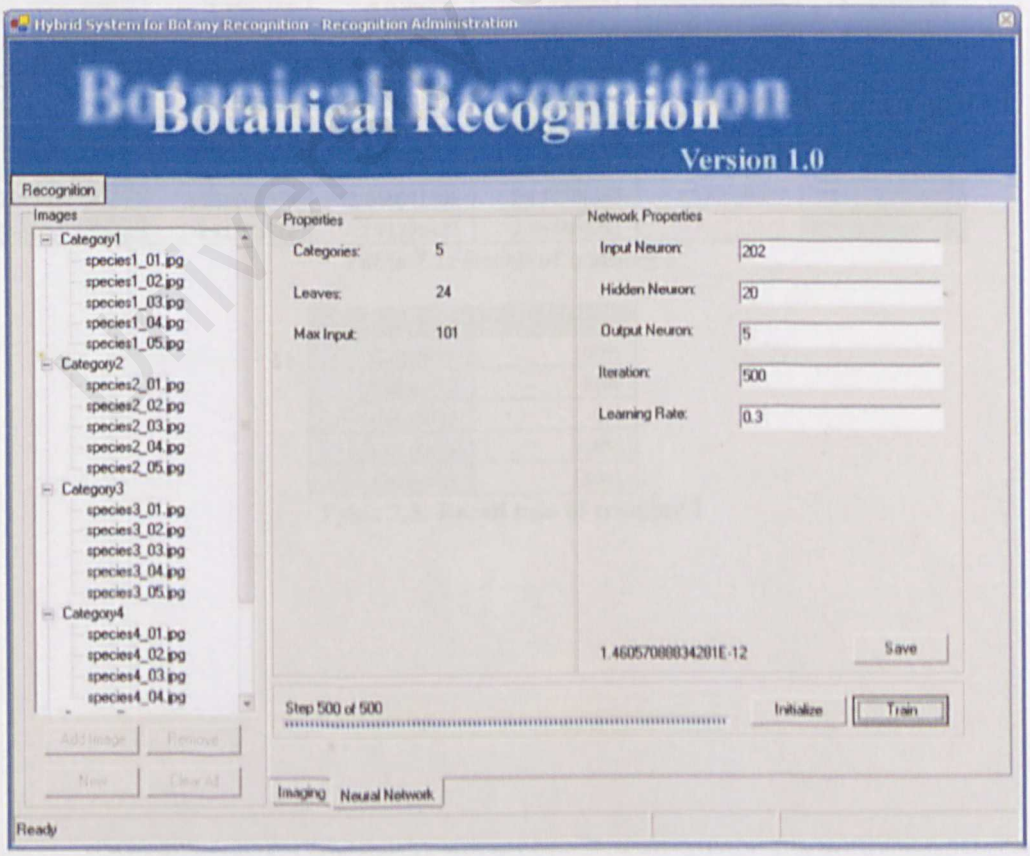



**Figure 7.19: Result of training 2**

All of the images in the training set are being used as recognition subject to test the recall of the trained neural network. Shaded column is the desired result while red color number is the recall result. Numbers that is nearer to 0 indicates the more similar it is to that category.

For training 1, the recall is as follows:

| Image | Similarity | | | | |
|---|---|---|---|---|---|
| | Category1 | Category2 | Category3 | Category4 | Category5 |
| Species1_01.jpg | 1.025823075 | 2.424461678 | 1.891622842 | 1.96519885 | 2.424461678 |
| Species1_02.jpg | 1.231688259 | 2.374132535 | 1.753543215 | 1.846771443 | 2.374132535 |
| Species1_03.jpg | 1.268436153 | 2.41790873 | 1.750147027 | 1.868737441 | 2.41790873 |
| Species1_04.jpg | 1.37489912 | 2.310045446 | 1.654477911 | 1.731158437 | 2.310045446 |
| Species1_05.jpg | 1.247684302 | 2.423860491 | 1.720838531 | 1.911155926 | 2.423860491 |
| Species2_01.jpg | 2.375164853 | 1.672367622 | 3.241396335 | 2.073696166 | 3.241396335 |
| Species2_02.jpg | 2.415874083 | 0.626069031 | 2.271219093 | 1.835627287 | 2.415874083 |
| Species2_03.jpg | 2.365233874 | 0.565883414 | 2.128820897 | 2.051430254 | 2.256372807 |
| Species2_04.jpg | 2.035207575 | 1.616833133 | 2.597509748 | 1.614078725 | 2.673878497 |
| Species2_05.jpg | 2.376915257 | 1.235580084 | 3.006357211 | 2.153085305 | 3.006357211 |
| Species3_01.jpg | 1.701704779 | 2.597613067 | 1.778114591 | 1.681244122 | 2.597613067 |
| Species3_02.jpg | 1.654083009 | 2.745981095 | 1.792129992 | 1.891297905 | 2.745981095 |
| Species3_03.jpg | 1.531726491 | 2.50694293 | 1.602753994 | 1.851683169 | 2.50694293 |
| Species3_04.jpg | 1.209441159 | 1.21715717 | 1.831426481 | 1.66908174 | 1.757706659 |
| Species3_05.jpg | 1.447622946 | 2.617785446 | 1.77022173 | 1.950069887 | 2.617785446 |
| Species4_01.jpg | 2.24886333 | 2.262295776 | 2.171414186 | 0.262295776 | 2.262295776 |
| Species4_02.jpg | 2.2206674 | 2.2206674 | 2.2206674 | 0.840979685 | 1.546609501 |
| Species4_03.jpg | 2.095327885 | 2.161905847 | 2.020078233 | 0.161905847 | 2.161905847 |
| Species4_04.jpg | 2.070023562 | 1.762177935 | 2.550312245 | 1.521668644 | 2.668004661 |
| Species5_01.jpg | 1.646980743 | 1.428263176 | 2.022382462 | 1.140536321 | 2.022382462 |
| Species5_02.jpg | 2.082837194 | 2.082837194 | 2.068693869 | 2.013446351 | 0.096748107 |
| Species5_03.jpg | 2.079845292 | 1.75528302 | 2.65752314 | 2.65752314 | 1.383376132 |
| Species5_04.jpg | 2.846041506 | 2.846041506 | 2.138381543 | 1.843639287 | 1.471262523 |
| Species5_05.jpg | 2.71296422 | 2.71296422 | 2.367864093 | 1.107235013 | 1.846630105 |

**Table 7.2: Recall of training 1**

| Category | Recall |
|---|---|
| Category1 | 100% |
| Category2 | 80% |
| Category3 | 0% |
| Category4 | 100% |
| Category5 | 60% |

**Table 7.3: Recall rate of training 1**

For training 2, the recall is as follows:

| Image | Similarity | | | | |
|---|---|---|---|---|---|
| | Category1 | Category2 | Category3 | Category4 | Category5 |
| Species1_01.jpg | 1.290820812 | 2.642771959 | 1.793348209 | 2.086389072 | 2.642771959 |
| Species1_02.jpg | 1.261458424 | 2.452422862 | 1.624919793 | 1.864672149 | 2.452422862 |
| Species1_03.jpg | 1.358267511 | 2.597777515 | 1.68904588 | 2.026297144 | 2.597777515 |
| Species1_04.jpg | 1.329284777 | 2.412815344 | 1.542237796 | 1.827153698 | 2.412815344 |
| Species1_05.jpg | 1.347517396 | 2.593780056 | 1.658516662 | 2.05201783 | 2.593780056 |
| Species2_01.jpg | 2.553325803 | 0.558555315 | 2.447745616 | 1.968559597 | 2.553325803 |
| Species2_02.jpg | 2.479103891 | 0.540099768 | 2.336005924 | 1.927610605 | 2.479103891 |
| Species2_03.jpg | 2.421399331 | 0.684349717 | 2.213824893 | 1.863741061 | 2.410052235 |
| Species2_04.jpg | 2.032888082 | 0.058066581 | 2.027767663 | 1.965011762 | 2.031758379 |
| Species2_05.jpg | 2.528365624 | 1.107627163 | 2.696123706 | 1.522050746 | 2.696123706 |
| Species3_01.jpg | 1.758945728 | 2.8180768 | 1.518643756 | 2.13024204 | 2.8180768 |
| Species3_02.jpg | 1.741498261 | 2.725619409 | 1.4719054 | 2.006307985 | 2.725619409 |
| Species3_03.jpg | 1.800279726 | 2.857150003 | 1.554030161 | 2.089115695 | 2.857150003 |
| Species3_04.jpg | 1.720983693 | 2.836515827 | 1.58067135 | 2.145899075 | 2.836515827 |
| Species3_05.jpg | 1.755274275 | 2.839747826 | 1.552163715 | 2.151682798 | 2.839747826 |
| Species4_01.jpg | 2.177859257 | 2.260606958 | 2.260606958 | 0.260606958 | 2.260606958 |
| Species4_02.jpg | 2.037513587 | 2.05598866 | 2.049210198 | 0.05598866 | 2.053898406 |
| Species4_03.jpg | 2.000000329 | 2.000000265 | 2.000000329 | 3.29E-07 | 2.000000128 |
| Species4_04.jpg | 2.000000058 | 2.000000058 | 1.999999975 | 1.73E-07 | 2.000000058 |
| Species5_01.jpg | 2.150816574 | 1.434157966 | 2.351906717 | 2.351906717 | 1.071735451 |
| Species5_02.jpg | 2.000000165 | 2.00000018 | 2.00000018 | 2.000000126 | 1.80E-07 |
| Species5_03.jpg | 2.118380924 | 1.841303865 | 2.75679095 | 2.75679095 | 1.319852621 |
| Species5_04.jpg | 2.176361222 | 2.178001476 | 2.178001476 | 1.879243964 | 0.292089312 |
| Species5_05.jpg | 2.194194954 | 2.19552598 | 2.193420117 | 1.880530395 | 0.307682511 |

**Table 7.4: Recall of training 2**

| Category | Recall |
|---|---|
| Category1 | 100% |
| Category2 | 100% |
| Category3 | 100% |
| Category4 | 100% |
| Category5 | 100% |

**Table 7.5: Recall rate of training 2**

From the result tables, the less the total squared error the training has, the higher recall rate it obtains.

Next is testing of generalization of the neural network. Figure 7.20 shows the image list used as testing subjects for the neural network. In the image list there is image that is from the same category as the training sets, while the others are from different.

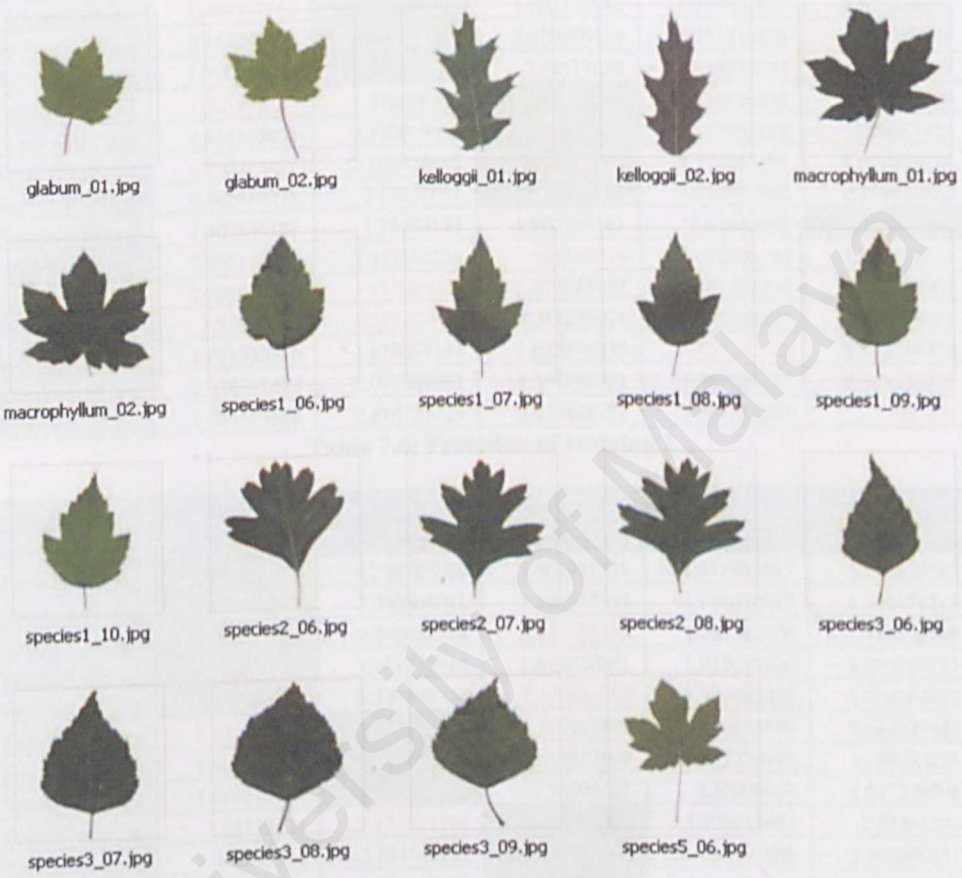



**Figure 7.20: Testing set**

| Image | Similarity | | | | |
|---|---|---|---|---|---|
| | Category1 | Category2 | Category3 | Category4 | Category5 |
| Species1_06.jpg | 1.180543163 | 2.078564189 | 1.510629274 | 1.637664252 | 2.078564189 |
| Species1_07.jpg | 1.012521244 | 2.093658528 | 1.15825945 | 1.995949711 | 2.087756827 |
| Species1_08.jpg | 1.901175835 | 2.253594574 | 1.24405881 | 1.861316414 | 1.924428919 |
| Species1_09.jpg | 1.192732675 | 1.978300431 | 1.219667399 | 1.866946806 | 2.047561262 |
| Species1_10.jpg | 1.543702436 | 2.222759767 | 1.384644694 | 2.012313083 | 1.78783815 |
| Species2_06.jpg | 1.489231163 | 2.097687583 | 1.922739916 | 1.999351959 | 2.624055862 |
| Species2_07.jpg | 2.143291736 | 0.919734553 | 2.429976106 | 2.211410878 | 1.891389841 |
| Species2_08.jpg | 1.790141586 | 1.227069408 | 2.28577635 | 2.28577635 | 1.491347157 |
| Species3_06.jpg | 1.307807132 | 1.370078559 | 1.834350036 | 1.597926811 | 1.999290329 |
| Species3_07.jpg | 1.641503831 | 2.370709952 | 1.210173859 | 2.17902869 | 2.09651429 |
| Species3_08.jpg | 1.826213801 | 2.105878692 | 1.235175786 | 1.773247891 | 1.699868574 |
| Species3_09.jpg | 1.402868373 | 2.17031651 | 1.316708021 | 1.718475962 | 2.158830362 |
| Species5_06.jpg | 1.902756181 | 1.79473174 | 1.902756181 | 1.61038568 | 0.558777306 |
| glabum_01.jpg | 2.335178245 | 2.335178245 | 1.541154914 | 1.631600728 | 1.576446434 |
| glabum_02.jpg | 2.322683127 | 1.157385077 | 2.322683127 | 2.047373294 | 1.541592902 |
| kelloggii_01.jpg | 1.74390801 | 1.337358818 | 2.281296921 | 1.352026525 | 2.281296921 |
| kelloggii_02.jpg | 1.051338566 | 1.816254141 | 1.622444236 | 1.429240809 | 1.893564354 |
| macrophyllum_01.jpg | 2.338761412 | 2.056084067 | 1.864580521 | 1.881939225 | 1.208155713 |
| macrophyllum_02.jpg | 1.044643886 | 1.816723026 | 2.073448751 | 1.661753203 | 2.22E+00 |

**Table 7.6: Precision of training 1**

| Image | Similarity | | | | |
|---|---|---|---|---|---|
| | Category1 | Category2 | Category3 | Category4 | Category5 |
| Species1_06.jpg | 1.269261481 | 2.220785285 | 1.467981735 | 1.617963567 | 2.171623887 |
| Species1_07.jpg | 1.197646393 | 2.050844041 | 1.386645594 | 1.659610587 | 1.996578311 |
| Species1_08.jpg | 1.80635291 | 2.398027648 | 1.532783253 | 1.865742826 | 1.911287186 |
| Species1_09.jpg | 1.418858267 | 1.811730974 | 1.639820261 | 1.452654926 | 2.080364731 |
| Species1_10.jpg | 1.429369675 | 2.270888954 | 1.379833665 | 1.999383023 | 1.832082012 |
| Species2_06.jpg | 1.172649107 | 1.975076574 | 1.760967829 | 1.70583799 | 2.246236482 |
| Species2_07.jpg | 1.995547652 | 1.17374543 | 2.39617199 | 2.142725799 | 1.769233225 |
| Species2_08.jpg | 1.465512309 | 1.416185791 | 2.30006967 | 2.30006967 | 1.547228108 |
| Species3_06.jpg | 1.65538124 | 1.176102494 | 2.169972393 | 1.706611607 | 2.27561594 |
| Species3_07.jpg | 1.616745141 | 2.304444285 | 1.282172997 | 1.96791802 | 2.015800831 |
| Species3_08.jpg | 1.91635521 | 2.206455775 | 1.254003974 | 1.731426416 | 1.703000852 |
| Species3_09.jpg | 1.549673568 | 2.096053595 | 1.438289011 | 1.29897 | 2.080148226 |
| Species5_06.jpg | 1.983495535 | 1.885133075 | 2.029943973 | 1.743834486 | 0.579641069 |
| glabum_01.jpg | 2.050653852 | 2.065893961 | 1.247208385 | 1.436240954 | 1.631044083 |
| glabum_02.jpg | 2.501712184 | 1.36362883 | 2.502781958 | 1.794995641 | 1.588777379 |
| kelloggii_01.jpg | 1.867323609 | 1.527715713 | 2.407300093 | 1.129547746 | 2.407300093 |
| kelloggii_02.jpg | 1.051024862 | 2.093180527 | 1.690728428 | 1.381308411 | 2.093180527 |
| macrophyllum_01.jpg | 2.347007475 | 2.057894999 | 2.177745243 | 1.656117827 | 0.885565086 |
| macrophyllum_02.jpg | 1.187188661 | 2.127451453 | 2.319664738 | 1.429588385 | 2.46E+00 |

**Table 7.7: Precision of training 2**

From the result table, neural network training 2 has a slightly better precision than the network training 1.

# CHAPTER 8: SYSTEM EVALUATION & CONCLUSION

## 8.1 *System Evaluation*

From the testing phase, the system has performed well. The edge detection algorithm gives a clear edge image from the input image. On the other hand, although the thinning algorithm cost much of processing power, it guarantees a one pixel edge output. Unfortunately the system has less features than propose. Only shape feature can be extracted from the input image. This is due to insufficient time to build the extra features.

The neural network perform well. With the Nguyen-Widrow weight initialization and modified targets, the neural network converge using less epoch than normal random initialization. However the inconsistent total squared error for each training makes the application of binary or bipolar almost no effects to the training convergence. Overall the neural network has a good recall rate if the total squared error drops below 0.0001. however the neural network show inconsistent precision on generalization.

On the other hand the system is still not user friendly. The input digital image has to be manually preprocessed so that it can be used in this system. The image processing and neural network configuration still difficult to be understand by the user. The current version of system also has less features than suppose to be.

## *8.2 Obstacle Faced*

**1. During Analysis Phase**

❖ Determining Scope of the system

Since there was no prior experience in developing a system, it was hard to determine to which extent to define the scope of the system so that it can be completed within the given time frame. Addition to using unfamiliar development language, the scope has been reduced so that a working version of system but has less features can be produced.

**2. During Requirement Phase**

❖ Time Constraint

During the requirement phase, there was not enough time to study and analyze the available solution of design in first semester. This was due to inexperience and insufficient knowledge of designing a system.

**3. During Implementation Phase**

❖ Problems on implementing algorithm

During implementation phase, some of the algorithm doesn't work as it should be. Alternative solution need to be search and replace and the time is insufficient to test thoroughly.

❖ Problems on development language

Although the development language used is easy to understand, implementing it onto the system still faces some obstacle. In addition to time limit using the faculty laboratory facility, solution are hard to acquire.

**4. During Testing Phases**

❖ Time Constraint

Due to large portion of time is used during implementation phase, testing of the system is not completely done well. Testing subjects is also limited due to unavailable of source as promised.

## 8.3 *System Strength*

- ✓ **User-friendly graphical interfaces.**

  This system is design with drag and click graphical interfaces. Most of the operation can be done by click and point operations. Thus, users will spend most of their time using the system without much input effort.

- ✓ **Flexibility**

  The system is design to allow flexibility of usage. Most of the feature can be use in many ways. The system coding is design in such a way so that each class can be reuse in other system without changing much of the coding. Although the design cost in reduce of performance time it doesn't affect much of the overall system results.

- ✓ **Optimize the performance**

  The system uses algorithm that optimize the performance of image processing and neural network. The overall system performance has increase from the previous version that doesn't implement the algorithm.

## *8.4 System Limitation*

- **Reduced scope**

  Due to limit of time in implementing the system, some of the purposed feature is not available in this system. The system only processes the shape but not the color and structure of the leaf.

- **Limited image preprocess function**

  The image processing function has less ability to preprocess the input image. This has resulted less flexibility in using other input image. Input image has to be static in background and oriented in the same way. Users have to manually preprocess the image before using as an input to the system.

- **Limited workspace saving function**

  Current version of system only allows users to save the trained neural network. The image processing module still lack of saving ability due to unidentified coding problems arises in the projects.

- **Thinning algorithm is processing intensive**

  Iterations of thinning algorithm take most of the processing time.

## 8.5 *Future Enhancement*

This system will provide a functional framework from which to evolve. Here are some future enhancement's suggestions:

- ✓ Feature like structure of the leaf should be considered add in to enhance the performance of this module.
- ✓ Better preprocessing function so that no manual editing is required
- ✓ Enhance the coding design to take less in processing time
- ✓ Token fetching method should be change to relative cosinus and sinus from the stem, not cosinus and sinus of two points. This makes the orientation of the leaf to be less affective to the result.

## 8.6 *Project Conclusion*

It was thought that this project would be able to determine if a neural network can be an effective tool for determining the plant species thought the leaf image analysis. Unfortunately, not enough research is being made to this project to know whether or not the neural network could return the better result or not. Although the two neural networks perform well in memorizing the training pattern, neither of them shows a constant result in generalizing the pattern. This will become worst is the input image is different to the training set. The resolution of the image also will affect the analysis and recognition.

Color feature cannot be used in analyzing because the color will varies on the photo taking process. Same leaf be analyze as different color if the flash angle is different.

Based on the results from the testing, it can be concluded that the neural network is very good in matching patterns that is fairly resemble to each other, but not an effective tools to be use as proposed objectives. However the testing should be conducted with more testing subjects with more variety of shapes to determine the effectiveness of the system.

# REFERENCES

[1].Yong, R., Huang, Thomas S., Chang, S.F. Image Retrieval: Current Techniques, Promising Directions And Open Issues.

[2].Kien, A. Hua, Khanh, Vu, Oh, J.Hwan. SamMatch: A Flexible and Efficient Sampling-Based Image Retrieval Technique for Large Image Databases.

[3].Ooi, B. Chin, Tan, K.-Lee, Chua, T. Seng, Wynne Hsu. Fast image retrieval using color-spatial information

[4].J.Bigun, J. M. H. Du Buf. Texture Segmentation By Real and Complex Moments of the Gabor Power Spectrum. Progress in Image Analysis and Processing II, pages 191-198, Switzerland. 1992

[5].John R Smith and Shih-Fu Chang. Automated binary texture feature sets for image retrieval. May 1996

[6].Apostol Natsev, Rajeev Rastogi and Kyuseok Shim. WALRUS: A Similarity Retrieval Algorithm for Image Databases.

[7].N. Vujovic and D. Brzakovic Evaluation of an Algorithm for finding a Match of a Distorted Texture pattern in a Large Image Database. Lehigh University.

[8].Asim bhatti, Saeid Nahavandi and Hong Zheng. Image Matching using TI Milt-Wavelet Transform. VII$^{th}$ Digital Image Computing: Techniques and Applications. Sydney.10-12 Dec, 2003.

[9].Jia Li, James Z. Wang and Gio Wiederhold. IRM: Integrated Region Matching for Image Retrieval. CA USA.

[10]. Farzin Mokntarian and Sadegh Abbasi. Matching Shapes With Self Intersections: Application to Leaf Classification. IEEE Transactions On Image Processing, Vol 13, No. 5, May 2004.

[11]. Sadegh Abbasi and Farzin Mokhtarian. Robustness of Shape Similarity Retrieval under Affine Transformation. England

[12]. A. Bollini, U. Cei, L. Lombardi. Using Neural Networks To Classify Objects. Progress in Image Analysis and Processing II. Pavia Italy.

[13]. Graham Seabrook. Real World Object Recognition With A neural Network. Progress in Image Analysis and Processing II. England.

[14]. Ashit Gandhi. Content Based Image Retrieval: Plant Species Identification.

[15]. Hyong K. Lee and Suk I. Yoo. A Neural Network-based flexible Image Retrieval. Korea.

[16]. Chih-Fong Tsai. Stacked Generalization: A Novel Solution to Bridge the Semantic Gap for Content-Based Image Retrieval. UK

[17]. Chih- Fong Tsai, Ken McGarry and John Tait. Image Classification Using Hybrid Neural Networks.

[18]. Dengsheng Zhang, Guojun Lu. Review of shape representation and description techniques.

[19]. Javad Haddadnia, KarimFaez, Majid Ahmadi. N-Feature Neural Network Human Face Recognition

[20]. M. Egmont-Petersena;, D. de Ridderb, H. Handelsc. Image processing with neural networks—a review.

[21]. Martin Lefley, Tom Kinsella. Investigating neural network efficiency and structure by weight investigation.

[22]. Scott E Umbaugh. Computer Vision and Image Processing: A Practical Approach Using CVIPtools. Prentice Hall PTR. 1999.

[23]. Jeffrey, L. Whitte, Lonnie, D. Bentley, Kevin, C. Dittman. System Analysis and Design Methods. 6th Edition. McGraw Hill. 2004.

[24]. Stephen, R. Schach. Object Oriented & Classical Software Engineering. 6th Edition. McGraw Hill.

[25]. Fausett, Laurene V. Fundamentals of neural networks: architectures, algorithms and applications.

[26]. EDGE / LINE DETECTION
http://ari.cankaya.edu.tr/~reza/ImLab4.htm

[27]. Overview of Neural Networks
http://www.benbest.com/computer/nn.html

[28]. Operating System Comparisons
http://www.commercialventvac.com/~jeffs/OS_comparison.html

[29]. Windows XP Professional Comparison Guide
http://www.microsoft.com/windowsxp/pro/evaluation/whyupgrade/featurecomp.mspx

[30]. VB.NET vs Java
http://www.theopensourcery.com/vbjava2.htm

[31]. Line Thinning
http://ct.radiology.uiowa.edu/~jiangm/courses/dip/html/node98.html

[32]. Morphology - Thinning
http://homepages.inf.ed.ac.uk/rbf/HIPR2/thin.htm

[33]. Robotics India - Your Online Robotics Community - Image Recognition for Robots - II
http://www.roboticsindia.com/modules.php?name=News&file=article&sid=64

[34]. Skeletonization
http://www.inf.u-szeged.hu/~palagyi/skel/skel.html

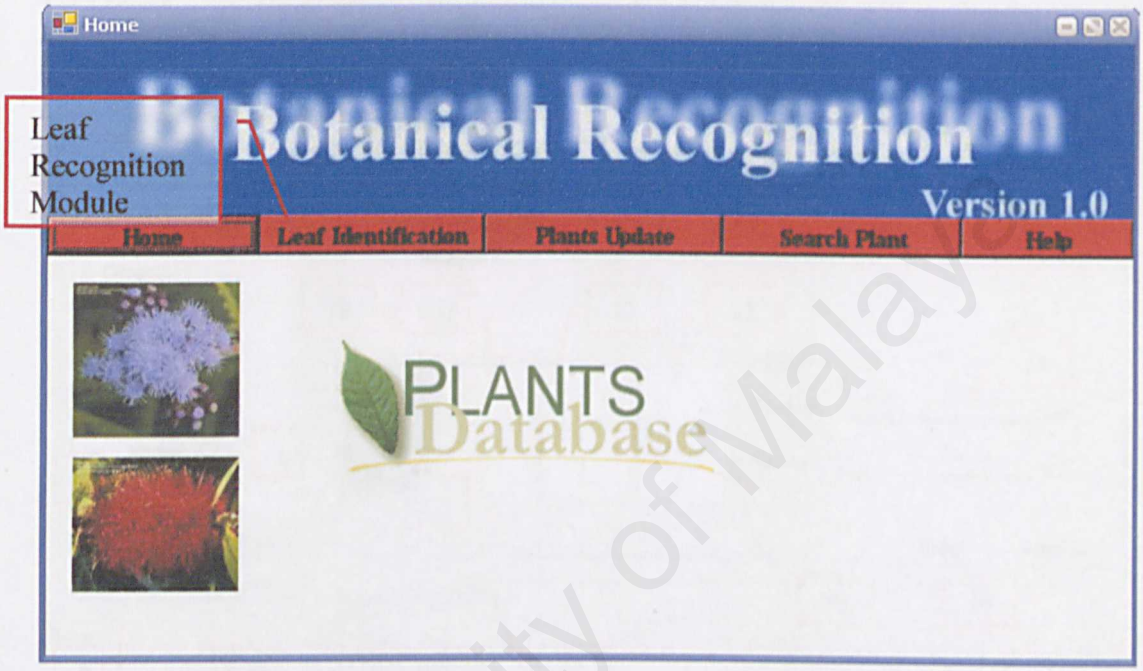
# APPENDIX

# USER MANUAL

## HYBRID SYSTEM FOR BOTANY RECOGNITION

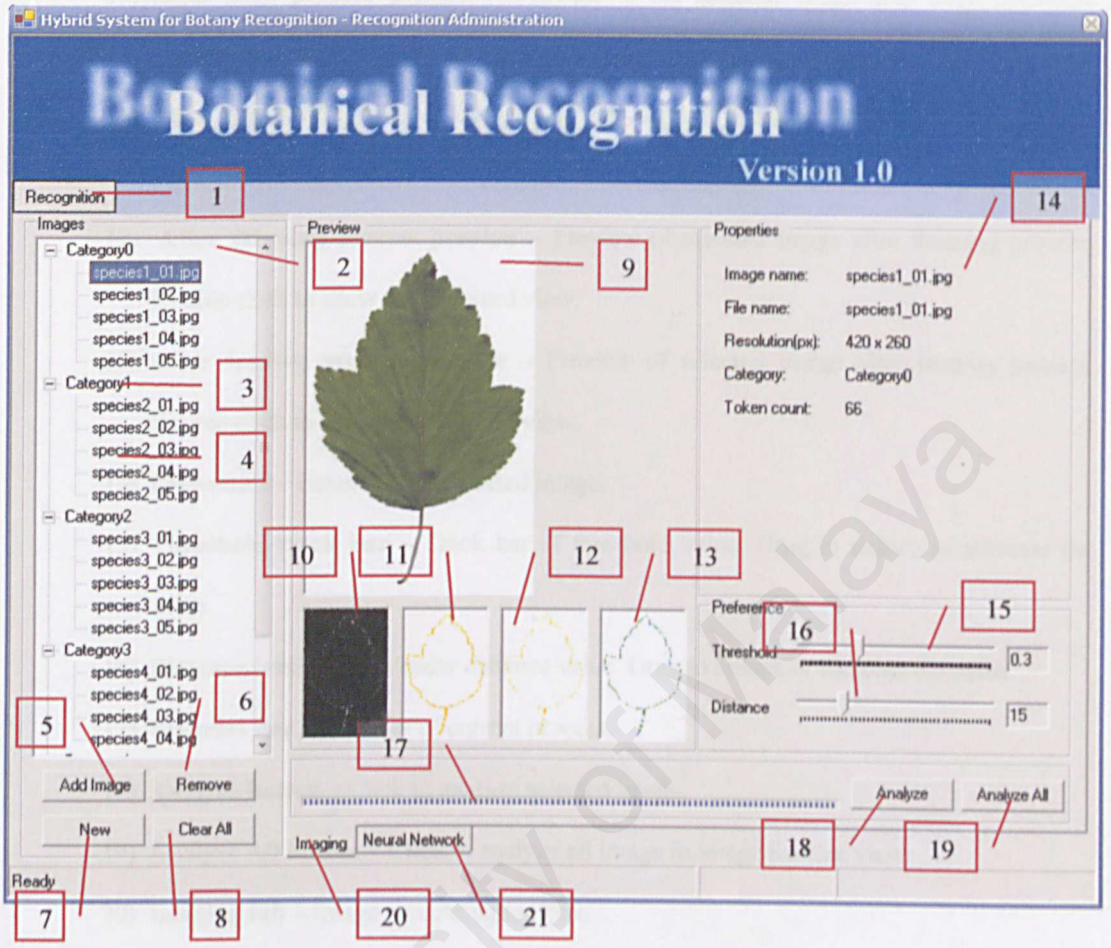
### A. SYSTEM COMPONENT

MAIN PAGE:



## LEAF RECOGNITION MODULE:

IMAGING TOOLBOX:




1) **Recognition button** – Click to go to leaf recognition toolbox.
2) **Images list tree view** – Shows list of images added into the system.
3) **Category nodes** – Shows the category added to the image list tree view. Click twice to rename the category.
4) **Image nodes** – Shows the images added to the system under corresponding category. Click twice to rename the image name. Drag to desired category node to change category.
5) **Add Image button** – Click to browse for image to add into image list.
6) **Remove button** – Click to remove selected node.
7) **New Button** – Click to add new category node.
8) **Clear All button** – Click to clear the image list tree view.

9) **Image Preview** – Preview of the selected image. Double click to show double sized view.

10) **After edge process preview** – Preview of the selected image after edge detection. Double click to show double sized view.

11) **After threshold preview** – Preview of selected image after threshold. Double click to show double sized view.

12) **After thinning process preview** – Preview of selected image after thinning process. Double click to show double sized view.

13) **After feeding process preview** – Preview of selected image after feeding process. Double click to show double sized view.

14) **Properties** – Details of the selected image.

15) **Threshold track bar** – Track bar of threshold value. Drag to reduce or increase the value.

16) **Distance track bar** – Feeder distance value. Drag to reduce or increase the value.

17) **Progress bar** – Progress of current process.

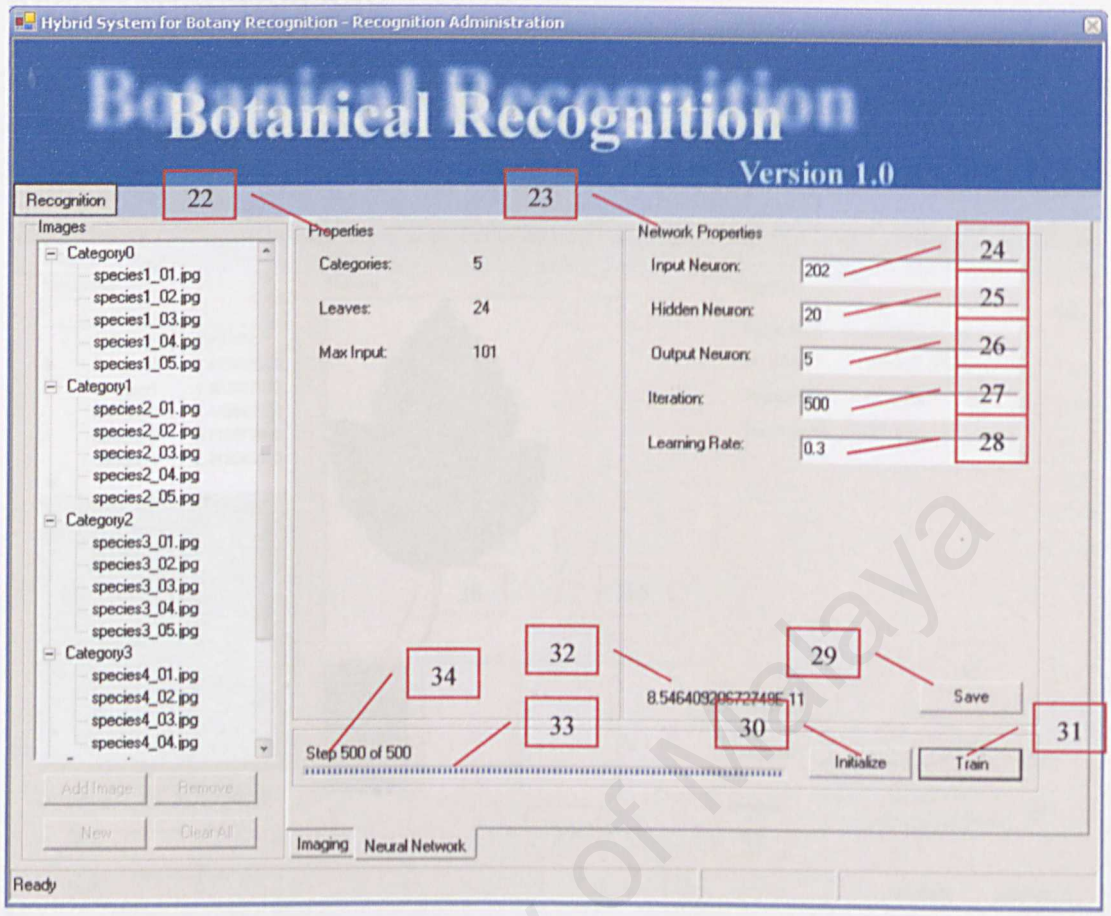
18) **Analyze button** – Click to analyze selected image.

19) **Analyze All button** – Click to analyze all image in image list tree view.

20) **Imaging tab** – Image processing toolbox.

21) **Neural Network tab** – Neural Network toolbox.

## NEURAL NETWORK TOOLBOX:




22) **Image Analysis properties** – Details of the image analysis process.

23) **Network Properties** – Neural network configurations.

24) **Input Neuron** – Amount of input neurons.

25) **Hidden Neuron** – Amount of hidden neurons.

26) **Output Neuron** – Amount of output neurons.

27) **Iterations** – Amount of training iterations.

28) **Learning Rate** – Value for the neural network learning rate.

29) **Save button** – Click to save the trained neural network. This button only available after the neural network training is complete.

30) **Initialize button** – Initialize the default neural network configurations.

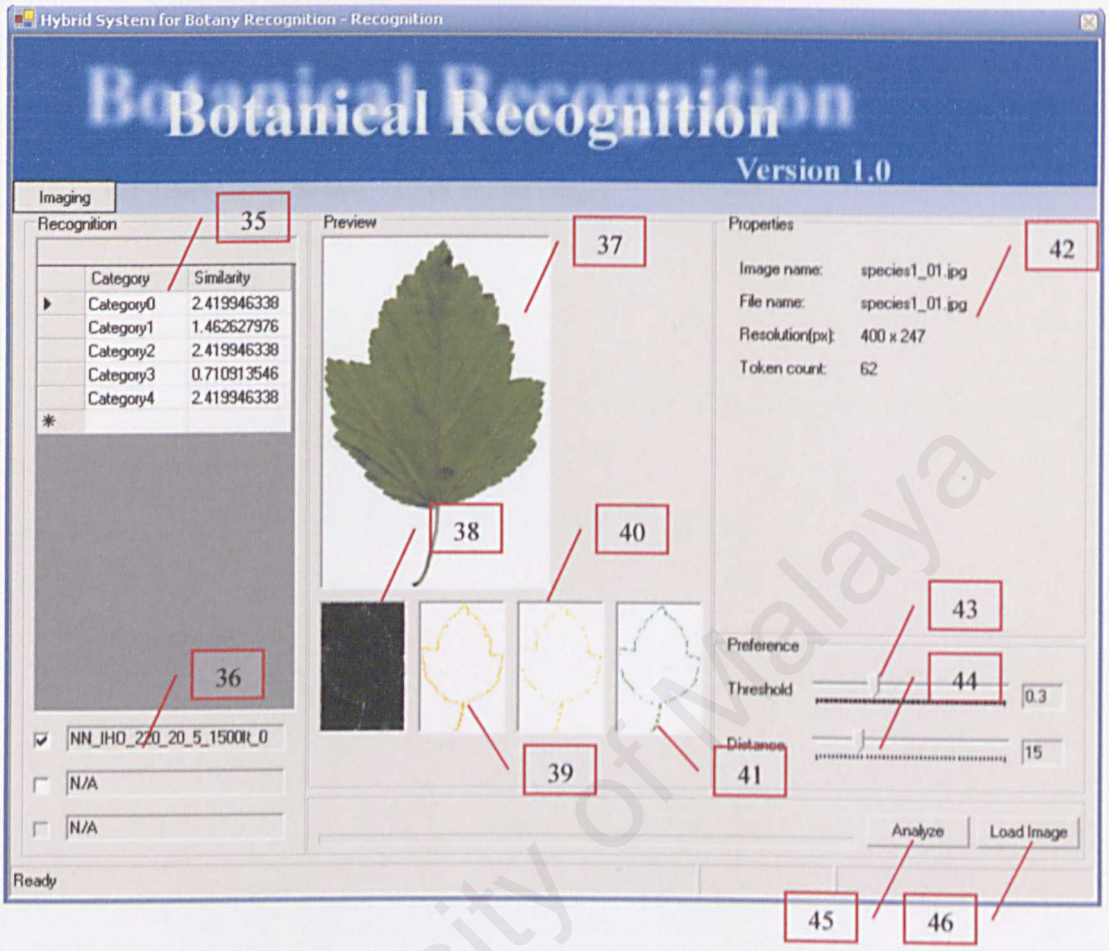
31) **Train button** – Train the neural network using the given configurations.

32) **Total Squared Error** – Total squared error for current iteration.

33) **Progress bar** – Training progress bar.

34) **Training Steps** – Current of total training steps.

RECOGNITION TOOLBOX:




35) **Result table** – Result of the recognition for the loaded image. Click to view the selected details

36) **Neural Net** – Click to load a saved neural net. Up to three neural net can be loaded.

37) **Image Preview** – Preview of the loaded image. Double click to show double sized view.

38) **After edge process preview** – Preview of the loaded image after edge detection. Double click to show double sized view.

39) **After threshold preview** – Preview of loaded image after threshold. Double click to show double sized view.

40) **After thinning process preview** – Preview of loaded image after thinning process. Double click to show double sized view.

41) **After feeding process preview** – Preview of loaded image after feeding process. Double click to show double sized view.

42) **Properties** – Details of the loaded image.

43) **Threshold track bar** – Track bar of threshold value. Drag to reduce or increase the value.

44) **Distance track bar** – Feeder distance value. Drag to reduce or increase the value.

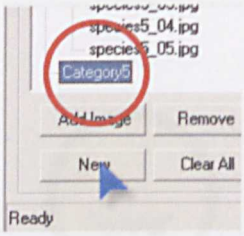
45) **Analyze button** – Click to analyze loaded image.

46) **Load Image button** – Click to load image.

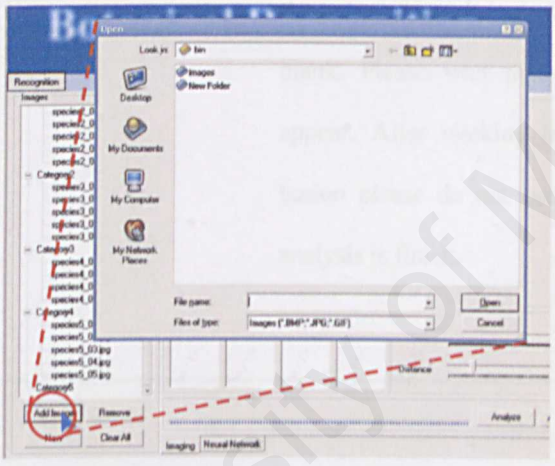
## B. USING THE SYSTEM

### a. ANALYZE IMAGE

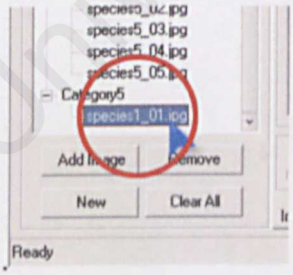
On the Imaging Toolbox, click on the 'New' (#7) button. A new category (#3) will be added into the image list tree view (#2).

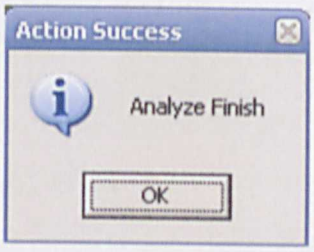



Select the category, and then click on 'Add Image' (#5) button. A dialog box will appear.



Browse for the image file to be added. Then click on the "Open" button on the dialog box. The image will be added into the previous selected category.




Adjust the threshold (#15) and distance (#16) value as desired. Select the image, and then click on the 'Analyze' (#18) button. An 'Action Success' popup message will appear after the image is analyzed.

Double click on any preview image to view zoom mode of the results.

Amount of tokens will be displayed in the properties (#14).

**b. ANALYZE ALL IMAGE**

Add all image that need to be analyze into the image list tree view according to their category. Then click on 'Analyze All' button. Wait until the 'Action Success' popup message appears. All images are ready to be used in the Neural Network Toolbox.

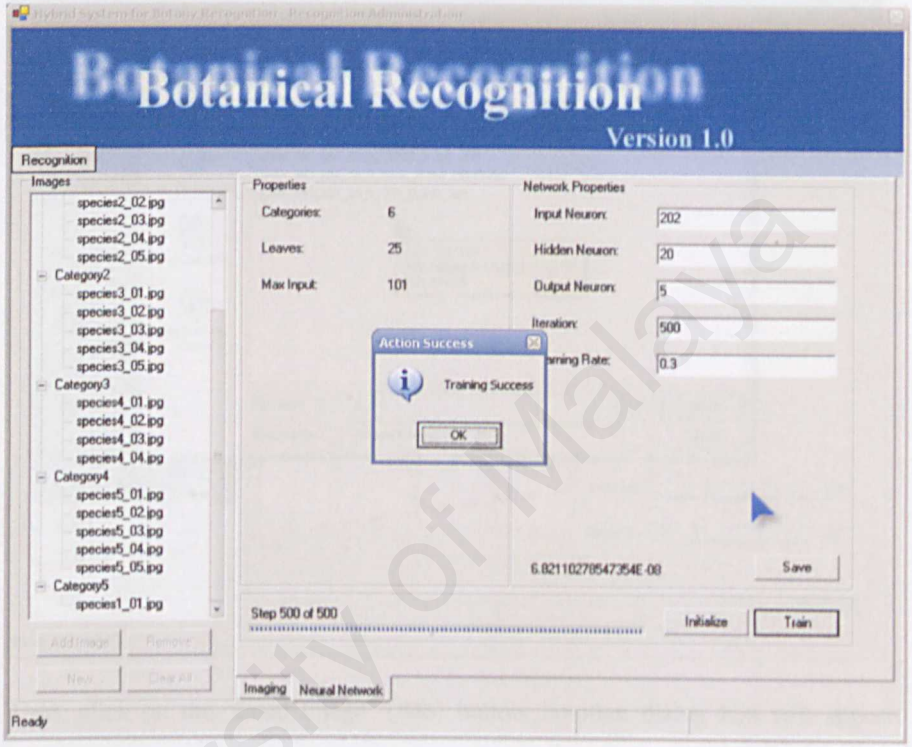
---


Analyzing image will cost a lot of processing power. The interface may shows 'Not Responding' most of the time or the progress bar doesn't move or the interface may appear blank. Please wait patiently until the popup windows appear. After clicking the 'Analyze' or 'Analyze All' button please do not perform other operations until the analysis is finish.

---

### c. TRAINING THE NEURAL NETWORK

In the Neural Network Toolbox, click on the 'Initialize' (#30) button to load the predefined neural network configurations. Make any necessary changes to the neural network settings. Then click on the 'Train' (#31) button. An 'Action Success' popup message will appear after the training is success. Click on 'Save' (#29) button to save the trained neural network.



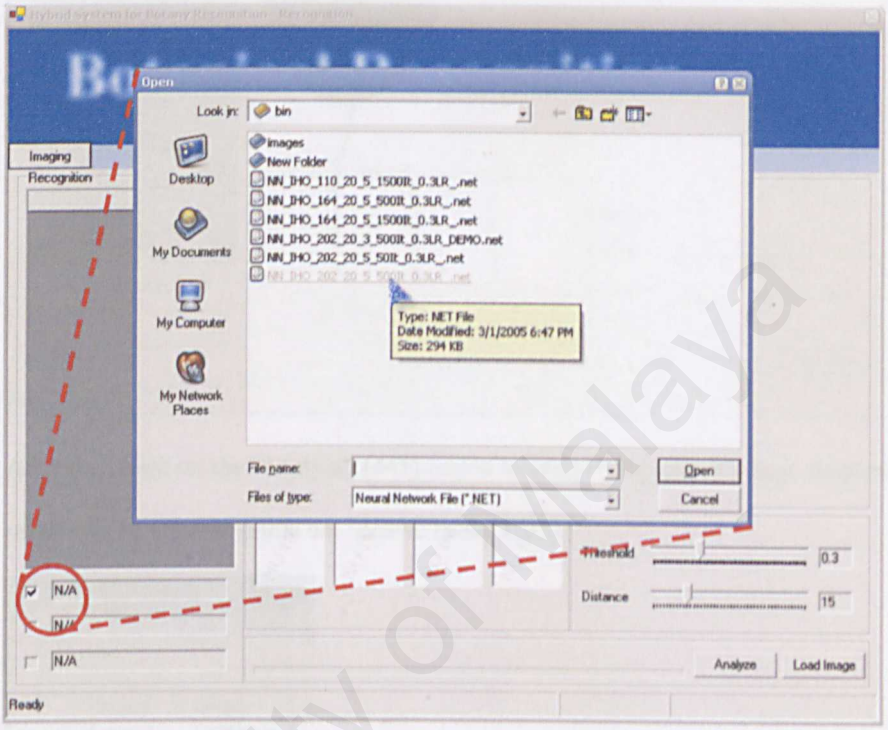


Training the neural network will cost a lot of processing power. The interface may shows 'Not Responding' most of the time or the progress bar doesn't move or the interface may appear blank. Please wait patiently until the popup windows appear. After clicking the 'Train' button please do not perform other operations until the training is finish.

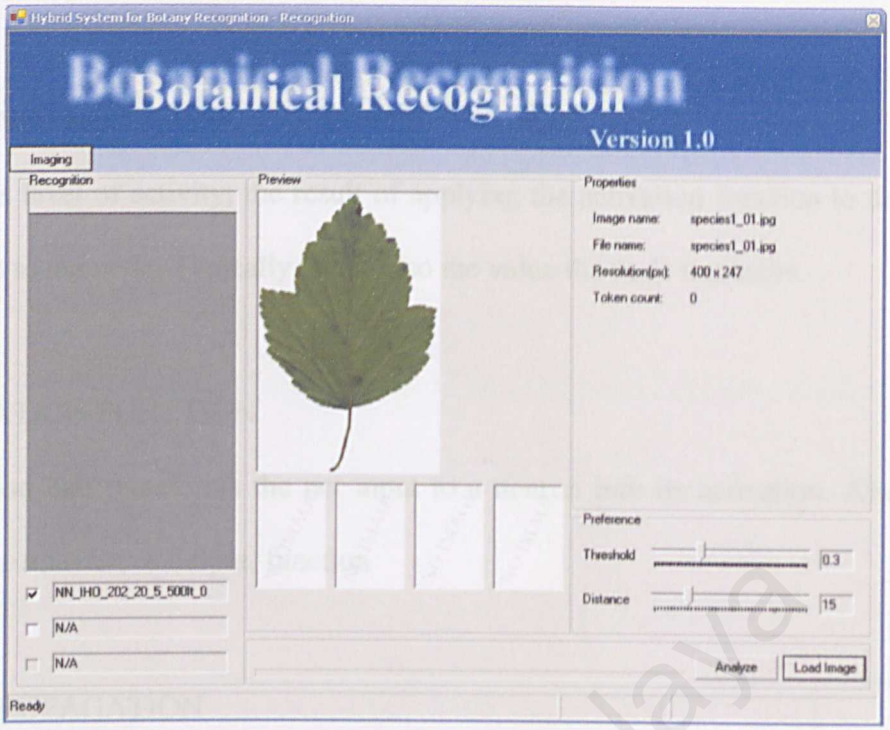
---

## d. RECOGNIZING AN IMAGE

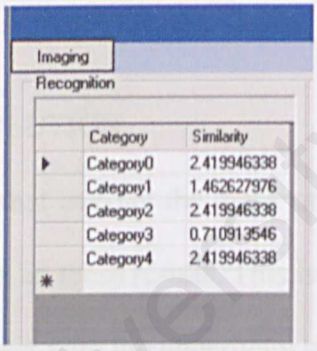
On the Recognition Toolbox, check on the first Neural Net check box (#36). A dialog box will appear. Browse for a neural net file. Click on the 'Open' button.




Then, click on the 'Load Image' (#46) button. Another dialog box will appear. Browse for the input image to be recognized. Click on the 'Open' button. The image will be loaded into the system.




After that, click on the 'Analyze' (#45) button to analyze the loaded image. Analysis result will be showed inside the 'Result Table' (#35).




Click on the row to view the details of the category.

# GLOSSARY

ACTIVATION

A node's level of activity; the result of applying the activation function to the net input to the node. Typically this is also the value the node transmits.

ACTIVATION FUNCTION

A function that transforms the net input to a neuron into its activation. Also known as transfer, or output, function

BACKPROPAGATION

A learning algorithm for multilayer neural nets based on minimizing the mean, or total, squared error.

CONVOLUTION

Process that overlay the mask on the image, multiplies the coincident values, and sums all these results. This is equivalent to finding the vector inner product of the mask with the underlying subimage.

FEEDFORWARD

A neural net in which the signals pass from the input units to the output units (possibly through intermediate layers of hidden units) without any connections back to previous layers.

GENERALIZATION

The ability of a neural net to produce reasonable responses to input patterns that is similar, but not identical to training patterns. A balance between memorization and generalization is usually desired.

IMAGE PREPROCESS

Initial processing on digital image primary to reduce data and make analysis task easier.

ITERATION

One performance of a calculation (or group of calculation) that must in general, be repeated several times. In the neural network literature this term may be used to mean an 'epoch' or a 'learning trial'.

LEARNING RATE

A parameter that controls the amount by which weights are changed during training.

MEMORIZATION

The ability to recall perfectly a pattern that has been learned.

NEURON

The computational unit in a neural network. Each processing element (PE) receives input signals from one or more other PEs, typically multiplied by the weight on the connection between the sending PE and the receiving PE. This weighted sum of the inputs is transformed into the PEs activation by the

activation function. The PEs output signal (its activation) is then sent on to the other PEs or used as output from the net.

PRUNING

Pruning is a process in which the branches of a skeleton in an image are removed. One end pixel of each branch is removed for each iteration in an iterative process. A problem with pruning is that it also removes an end pixel from the object, which is wanted, unless it is a closed line. This means that the pruning for not closed objects shall be restricted to a few iterations to avoid removing too much of the wanted information.

THINNING

Thinning is a process in which the outer layer of a component in an image is removed. The process is done by applying masks to the outer layer of the component. If the mask fits the pixel in the middle is removed and the process continues until all pixels at the surface of the component have been tried with the mask. This means that all the outer pixels have to be identified and thereafter tried with the mask.

THRESHOLD

A value used in some activation functions to determine the units output. Mathematically the effect of changing the threshold is to shift the graph of the activation function to the right or left; the same effect can be accomplished by including a bias.

TOTAL SQUARED ERROR

Used in stopping conditions for backpropagation training. The squared error is the summed over all output components and over all training patterns.

WEIGHT

A value associated with a connection path between two processing elements in a neural network. It is used to modify the strength of a transmitted signal in many networks.